

# Reasoning about Independence in Probabilistic Models of Relational Data

Marc Maier

MAIER@CS.UMASS.EDU

Katerina Marazopoulou

KMARAZO@CS.UMASS.EDU

David Jensen

JENSEN@CS.UMASS.EDU

*School of Computer Science*

*University of Massachusetts Amherst*

*Amherst, MA 01003, USA*

Editor:

## Abstract

Bayesian networks leverage conditional independence to compactly encode joint probability distributions. Many learning algorithms exploit the constraints implied by observed conditional independencies to learn the structure of Bayesian networks. The rules of  $d$ -separation provide a theoretical and algorithmic framework for deriving conditional independence facts from model structure. However, this theory only applies to Bayesian networks. Many real-world systems, such as social or economic systems, are characterized by interacting heterogeneous entities and probabilistic dependencies that cross the boundaries of entities. Consequently, researchers have developed extensions to Bayesian networks that can represent these relational dependencies. We show that the theory of  $d$ -separation inaccurately infers conditional independence when applied directly to the structure of probabilistic models of relational data. We introduce relational  $d$ -separation, a theory for deriving conditional independence facts from relational models. We provide a new representation, the *abstract ground graph*, that enables a sound, complete, and computationally efficient method for answering  $d$ -separation queries about relational models, and we present empirical results that demonstrate effectiveness.

**Keywords:** relational models,  $d$ -separation, conditional independence, lifted representations, directed graphical models

## 1. Introduction

Bayesian networks are a widely used class of graphical models that are capable of compactly representing a joint probability distribution over a set of variables. The joint distribution can be factorized into a product of much smaller conditional distributions by assuming that variables are independent of their non-descendants given their parents (the Markov condition). The Markov condition ties the structure of the model to the set of conditional independencies that hold over all probability distributions the model can represent. Accurate reasoning about such conditional independence facts is the basis for constraint-based algorithms, such as PC, FCI, and MMHC, that are commonly used to learn the structure of Bayesian networks (Spirtes et al., 2000; Tsamardinos et al., 2006). Under a small number

of assumptions and with knowledge of the conditional independencies, these algorithms can identify causal structure (Pearl, 2000; Spirtes et al., 2000).

Deriving the full set of conditional independencies implied by the Markov condition is complex, requiring manipulation of the joint distribution and various probability axioms. Fortunately, the exact same set of conditional independencies entailed by the Markov condition are also entailed by  $d$ -separation, a set of graphical rules that algorithmically derive conditional independence facts directly from the model structure. That is, the Markov condition and  $d$ -separation are equivalent approaches for producing conditional independence from Bayesian networks (Neapolitan, 2004). When interpreting a Bayesian network causally, the causal Markov condition (variables are independent of their non-effects given their direct causes) and  $d$ -separation have been shown to provide the correct connection between causal structure and conditional independence (Scheines, 1997).

Bayesian networks assume that data instances are independent and identically distributed, but many real-world systems are characterized by interacting heterogeneous entities. For example, social network data consist of individuals, groups, and their relationships; citation data involve researchers collaborating and authoring scholarly papers that cite prior work; and sports data comprise players, coaches, teams, referees, and their competitive interactions. Over the past 15 years, researchers in statistics and computer science have devised more expressive classes of directed graphical models, such as probabilistic relational models (PRMs), which remove the assumptions of independent and identically distributed instances to more accurately describe these types of domains (Getoor and Taskar, 2007). Relational models generalize models that incorporate interference, spillover effects, or violations of the stable unit treatment value assumption (SUTVA) (Hudgens and Halloran, 2008; Tchetgen and VanderWeele, 2012) and multilevel or hierarchical models (Gelman and Hill, 2007). Many practical applications have also benefited from learning and reasoning with relational models. Examples include analysis of gene regulatory interactions (Segal et al., 2001), scholarly citations (Taskar et al., 2001), and biological cellular networks (Friedman, 2004).

Relational models also assume the Markov condition for connecting the underlying joint distribution with the structure of the model. Just like Bayesian networks, knowledge of conditional independencies that hold over relational data is important for relational models. Relational data typically encompass a much larger space of variables; thus, the ability to factorize the joint distribution into a set of smaller conditional distributions is crucial for representation and inference. Furthermore, constraint-based algorithms for learning the structure of relational models, such as Relational PC (Maier et al., 2010), are more efficient than searching the entire model space, but they rely on conditional independence facts to discover model structure.

In this paper, we show that  $d$ -separation does not correctly produce conditional independence facts when applied directly to relational models. We introduce an alternative representation, the *abstract ground graph*, that enables an algorithm for deriving conditional independence facts from relational models. We show that this algorithm is sound, complete, and computationally efficient, and we provide an empirical demonstration of the effectiveness of our approach across synthetic causal structures of relational domains.

In Section 2, we present an example that highlights why standard  $d$ -separation is inadequate for relational models. Section 3 offers background on propositional (Bayesian

network) models, and Section 4 formalizes the fundamental concepts of relational data and models. In Section 5, we define relational  $d$ -separation and abstract ground graphs, and we present our soundness and completeness results. In Section 6, we evaluate how frequently  $d$ -separation on relational models is equivalent to  $d$ -separation on abstract ground graphs, and Section 7 contains additional experimental results. Finally, Section 8 discusses the implications that relational  $d$ -separation has for representation and learning and provides future directions for this work.

## 1.1 Why $d$ -separation is Useful

A Bayesian network, as a model of a joint probability distribution, enables a wide array of useful tasks by supporting inference over an entire system of variables. They have been successfully applied to many domains, ranging from bioinformatics and medicine to computer vision and information retrieval. Naïvely specifying a joint distribution by hand requires an exponential number of states; however, Bayesian networks represent joint probability distributions that can be factorized into a product of conditional probability distributions. Bayesian networks leverage the Markov condition to represent conditional independencies in order to compactly specify a joint probability distribution.

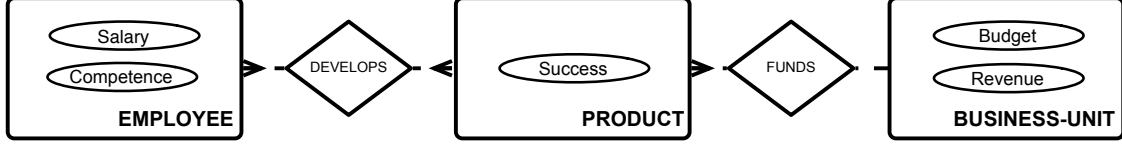
Alternative to the Markov condition, but equivalent in its implications,  $d$ -separation provides an algorithmic framework to derive the conditional independencies encoded by the model. These conditional independence facts are guaranteed to hold in every faithful joint distribution the model represents and, consequently, any data instance sampled from those distributions. The semantics of holding across all distributions is the main reason why  $d$ -separation is a useful theory.

Causal discovery, the task of learning generative models of observational data, superficially appears to be a futile endeavor. Yet learning and reasoning about the causal structure underlying real domains is a primary goal for many researchers. Fortunately,  $d$ -separation offers a connection between causal structure and conditional independence. The theory of  $d$ -separation can be leveraged to constrain the hypothesis space by eliminating models that are inconsistent with observed conditional independence facts. While many distributions do not lead to uniquely identifiable models, this approach (under simple assumptions) frequently discovers useful causal knowledge for domains that can be represented as a Bayesian network.

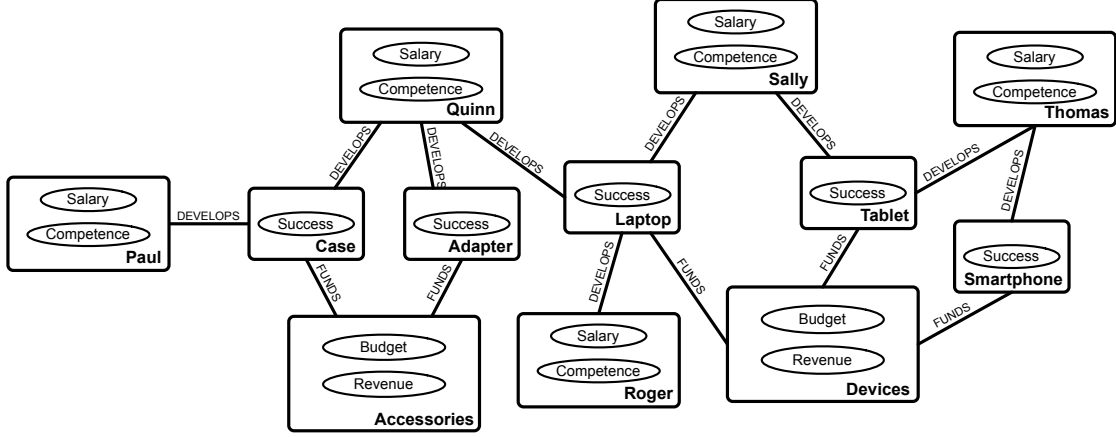
As described above, relational models more closely represent the real-world domains that many social scientists and other researchers investigate. These models can represent systems of interacting heterogeneous entities and probabilistic dependencies that cross entity boundaries. In order to successfully learn causal models of observational relational data, we need a similar theory for deriving conditional independence from relational models. In this paper, we formalize the theory of relational  $d$ -separation, which will provide a theoretical framework for algorithms that learn causal models of relational domains.

## 2. Example

Consider a corporate analyst who was hired to identify which products and employees are effective and productive for some organization. If the company is structured as a pure project-based organization, the analyst may collect data as described by the relational



(a) Example relational schema for an organization consisting of employees working on products, which are funded by specific business units within a corporation.

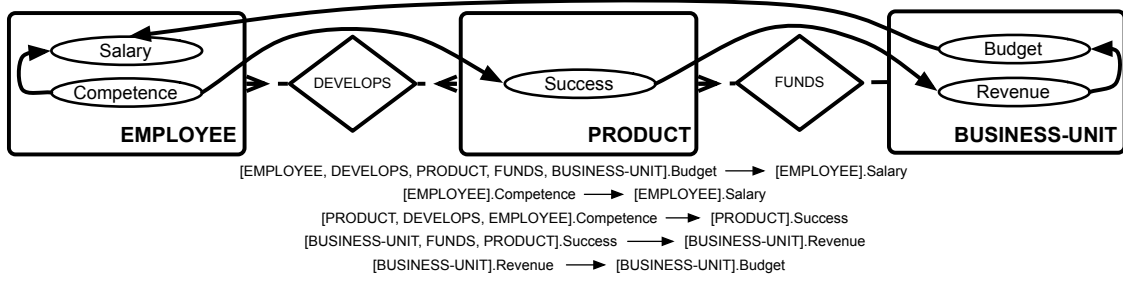


(b) Example fragment of a relational skeleton. Roger and Sally are employees, both of whom develop the Laptop product, but only Sally works on product Tablet. Both products Laptop and Tablet are funded by business unit Devices.

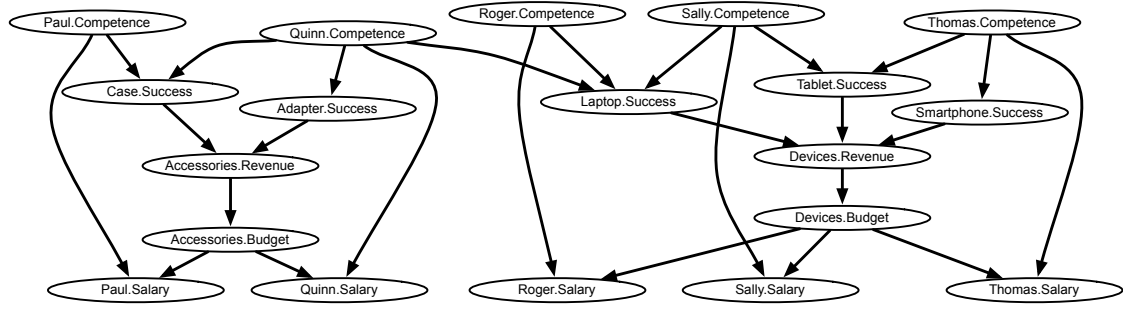
Figure 1: An example relational schema and skeleton for the organization domain.

schema in Figure 1(a). The schema denotes that employees can collaborate and work on multiple products, each of which is funded by a specific business unit. The analyst has also obtained variables on each entity—salary and competence of employees, the success of each product, and the budget and revenue of business units. In this example, the organization consists of five employees, five products, and two business units, which are shown in the relational skeleton in Figure 1(b).

The analyst may believe that the organization operates under the model depicted in Figure 2(a). For example, the competence of an employee affects the success of products they develop, and the revenue of a business unit is influenced by the success of products that it funds. The analyst then needs to verify the model structure in order to accurately advise executive decisions, such as determining which business units should have increased funding, which employees are being fairly or unfairly compensated, etc. Perhaps the analyst has experience in graphical models and decides to check that the conditional independencies encoded by the model are reflected in the data, assuming the faithfulness condition (see Section 3). The analyst then naïvely applies *d*-separation to the model structure in an attempt to derive these conditional independencies. However, applying *d*-separation directly to relational models does not correctly derive the set of conditional independencies. In other words, *d*-separation applied directly to relational models is not equivalent to the Markov condition.



(a) Example relational model. Competence of employees cause the success of products they develop, which in turn influences the revenue received by the business unit funding the product. Additional dependencies involve the budget of business units and employee salaries. The dependencies are specified by relational paths, listed below the graphical model.



(b) Example fragment of a ground graph. The success of product Laptop is influenced by the competence of both Roger and Sally. The revenue of business unit Devices is caused by the success of all its funded products—Laptop, Tablet, and Smartphone.

Figure 2: An example relational model and ground graph for the organization domain.

Naïvely applying  $d$ -separation to the model in Figure 2(a) suggests that employee competence is conditionally independent of the revenue of business units given the success of products. To see why this approach is flawed, we must consider the *ground graph*. A necessary precondition for inference is to apply a model to a data instantiation, which yields a ground graph to which  $d$ -separation can be applied. For a Bayesian network, a ground graph consists of replicates of the model structure for each data instance. In contrast, a relational model defines a template for how dependencies apply to a data instantiation, resulting in a ground graph with varying structure (see Section 4 for more details on ground graphs).

Figure 2(b) shows the ground graph for the relational model in Figure 2(a) applied to the relational skeleton in Figure 1(b). This ground graph illustrates that simply conditioning on product success can activate a path through the competence of other employees who develop the same products—we call this a *relational  $d$ -connecting path*.<sup>1</sup> Checking  $d$ -separation on

1. The indirect effect attributed to a relational  $d$ -connecting path is often referred to as interference, a spillover effect, or a violation of the stable unit treatment value assumption (SUTVA) because the treatment of one instance (employee competence) affects the outcome of another (the revenue of another employee’s business unit).

the ground graph indicates that to  $d$ -separate employee competence from business unit revenue, we must not only condition on the success of developed products, but also on the competence of other employees who work on those products (e.g.,  $\text{Roger.Competence} \perp\!\!\!\perp \text{Devices.Revenue} \mid \{\text{Laptop.Success}, \text{Sally.Competence}\}$ ).

This example also demonstrates that the Markov condition can be violated when directly applied to the structure of a relational model. In this case, the Markov condition applied to the model in Figure 2(a) implies that  $P(\text{Competence}, \text{Revenue} \mid \text{Success}) = P(\text{Competence} \mid \text{Success})P(\text{Revenue} \mid \text{Success})$ , that revenue is independent of its non-descendants (competence) given its parents (success). However, the ground graph shows the opposite, for example,  $P(\text{Roger.Competence}, \text{Devices.Revenue} \mid \text{Laptop.Success}) \neq P(\text{Roger.Competence} \mid \text{Laptop.Success}) P(\text{Devices.Revenue} \mid \text{Laptop.Success})$ . This is not surprising since the conditional independencies entailed by  $d$ -separation are known to be equivalent to those entailed by the Markov condition (Neapolitan, 2004). In fact, this is not the only conditional independence fact for which  $d$ -separation produces an incorrect result for this example model. Over 80% of the pairs of relational variables cannot be described by direct inspection of the model, and of those that can (such as the above example), 91% yield an incorrect conclusion. This is a single data point of a larger empirical evaluation presented in Section 6.

It might appear that, since the standard rules of  $d$ -separation apply to Bayesian networks and the ground graphs of relational models are also Bayesian networks, that applying  $d$ -separation to relational models is a non-issue. However, applying  $d$ -separation to a single ground graph may result in potentially unbounded runtime if the instantiation is large—especially given that relational databases can be arbitrarily large. Further, and more importantly, the semantics of  $d$ -separation require that conditional independencies hold across all possible model instantiations. Although  $d$ -separation can apply directly to a ground graph, these semantics prohibits reasoning about a single ground graph.

The conditional independence facts derived from  $d$ -separation hold for all faithful distributions represented a Bayesian network. Therefore, the implications of relational  $d$ -separation should analogously hold for all faithful distributions of variables for the space of all possible ground graphs. It is simple to show that  $d$ -separation holds for any ground graph of a Bayesian network—every ground graph consists of a set of disconnected sub-graphs, each of which has a structure that is identical to that of the model. However, relational models are templates, and the resulting ground graphs vary with the relational structure of the underlying data (e.g., different products are developed by varying numbers of employees). As a result, relational  $d$ -separation queries must be answered without respect to ground graphs. Additionally, the example illustrates how relational dependencies can exhibit  $d$ -connecting paths that are only manifest in ground graphs, not the model representation. In Section 5, we describe a new representation, the abstract ground graph, that can be used to reason about  $d$ -separation for relational models.

### 3. Background on Propositional Data and Models

A common assumption in classical statistics, machine learning, and causal discovery is that data instances are independent and identically distributed (IID). The first condition assumes that the variables on any given data instance are marginally independent of the variables

of any other data instances. The second condition assumes that every data instance is drawn from the same underlying joint probability distribution. IID data (also referred to as propositional data<sup>2</sup>) are effectively represented as a single table, where rows correspond to independent instances and columns are attributes of those instances.

The directed acyclic graph (DAG), or Bayesian network, is a widely used probabilistic graphical model of propositional data (Pearl, 1988). A DAG,  $G = (\mathbf{V}, \mathbf{E})$ , consists of a set of vertices  $\mathbf{V}$  corresponding to random variables in the data and a set of edges  $\mathbf{E} \subset \mathbf{V} \times \mathbf{V}$  encoding the conditional independencies expected to hold among the variables. Each variable  $V \in \mathbf{V}$  is described by a conditional probability distribution  $P(V \mid \text{parents}(V))$ , where  $\text{parents}(V) \subseteq \mathbf{V} \setminus \{V\}$  is the set of parent variables for  $V$ .

If the joint probability distribution  $P(\mathbf{V})$  satisfies the Markov condition for  $G$ , then  $P(\mathbf{V})$  can be compactly factorized according to the conditional distributions,  $P(\mathbf{V}) = \prod_{V \in \mathbf{V}} P(V \mid \text{parents}(V))$ . The Markov condition states that every variable  $V \in \mathbf{V}$  is conditionally independent of its non-descendants given its parents, where the descendants of  $V$  are all variables reachable by a directed path from  $V$ . Deriving the set of conditional independencies from  $G$  based on the Markov condition is cumbersome, requiring complex combinations of probability axioms. Fortunately,  $d$ -separation, a set of graphical criteria, provides the foundation for algorithmic derivation of all conditional independencies in  $G$  (Geiger et al., 1990) and entails the exact same set of conditional independencies as the Markov condition (Neapolitan, 2004).

In the following definition, a path is a sequence of vertices following edges in either direction. We say that a variable  $V$  is a collider on a path  $p$  if the two arrowheads point at each other (collide) at  $V$ ; otherwise,  $V$  is a non-collider on  $p$ .

**Definition 3.1 ( $d$ -separation)** Let  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $\mathbf{Z}$  be disjoint sets of variables in directed acyclic graph  $G$ . A path from some  $X \in \mathbf{X}$  to some  $Y \in \mathbf{Y}$  is  $d$ -connected by  $\mathbf{Z}$  if and only if every collider  $W$  on the path, or a descendant of  $W$ , is a member of  $\mathbf{Z}$ , and there are no non-colliders in  $\mathbf{Z}$ . Then, say that  $\mathbf{X}$  and  $\mathbf{Y}$  are  $d$ -separated by  $\mathbf{Z}$  if and only if there are no  $d$ -connecting paths between  $\mathbf{X}$  and  $\mathbf{Y}$  given  $\mathbf{Z}$ .

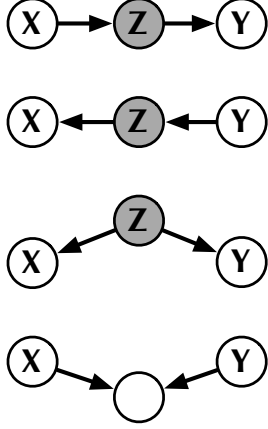
Figure 3(a) depicts the graphical patterns found along paths that lead to  $d$ -separation or  $d$ -connection based on Definition 3.1, and Figure 3(b) provides example  $d$ -separated and  $d$ -connected paths. At a first glance, identifying conditional independence facts from the rules of  $d$ -separation appears computationally intensive, testing a potentially exponential number of paths. However, Geiger et al. (1990) provide a linear-time algorithm based on breadth-first search and reachability on  $G$ .

Under a few assumptions, Bayesian networks can be interpreted causally, with edges corresponding to direct causal dependencies. If  $X \rightarrow Y$  is an edge in the causal model  $G$ , then manipulating or changing the value of  $X$  will alter the conditional distribution of  $Y$ — $P(Y \mid \text{do}(X))$  using Pearl’s *do*-calculus notation for interventions (Pearl, 2000). The causal interpretation of  $G$  assumes the *causal* Markov condition, which is identical to the Markov condition, replacing parents with direct causes and non-descendants with non-effects. In order for the causal Markov condition to hold, the variables  $\mathbf{V}$  must also be

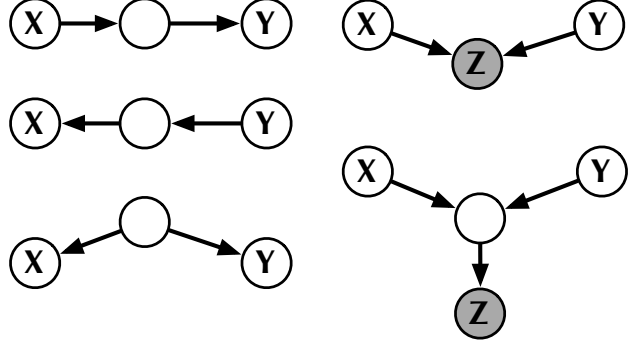
---

2. IID data are typically referred to as *propositional* because the data can be equivalently expressed under propositional logic.

***d*-separating path elements**  
(exists one on path)

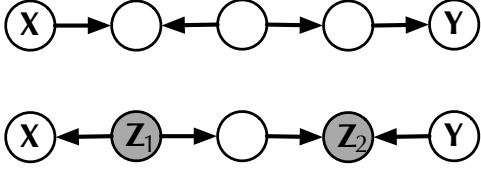


***d*-connecting path elements**  
(exists all on path)

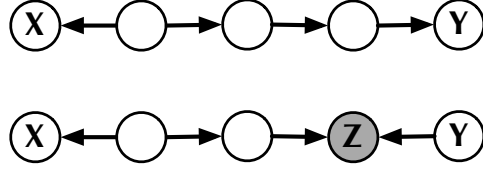


(a) Graphical patterns of *d*-separating and *d*-connecting path elements among disjoint sets of variables  $\mathbf{X}$  and  $\mathbf{Y}$  given  $\mathbf{Z}$ . Paths for which there exists a non-collider in  $\mathbf{Z}$  or a collider not in  $\mathbf{Z}$  are *d*-separating. Paths for which all non-colliders are not in  $\mathbf{Z}$  and all colliders (or a descendant of colliders) are in  $\mathbf{Z}$  are *d*-connecting.

***d*-separated paths**



***d*-connected paths**



(b) Several example *d*-separated and *d*-connected paths that illustrates the composition of path elements.

Figure 3: Patterns of *d*-separating and *d*-connecting path elements and example *d*-separating and *d*-connecting paths.

*causally sufficient*—there can be no latent common causes for any pair of variables in  $\mathbf{V}$ . The causal Markov condition is also equivalent to *d*-separation; therefore, both provide the connection between causal structures and probability distributions.

The conditional independencies entailed by both the causal Markov condition and *d*-separation hold over all *faithful* distributions that  $G$  can represent. A distribution  $P$  is faithful to  $G$  if and only if all conditional independencies in  $P$  that are entailed by the causal Markov condition on  $G$ . Additionally, if  $P$  is assumed to be faithful to  $G$ , then there are algorithms that can learn the Markov, or likelihood, equivalent set of causal models. That is, these algorithms correctly identify the causal dependencies in  $G$  that are consistent with observed conditional independence facts across  $\mathbf{V}$ . These algorithms typically exploit



the rules of  $d$ -separation and the assumption of model acyclicity to determine the direction of causality (Spirtes et al., 2000).

#### 4. Concepts of Relational Data and Models

Propositional representations describe domains with a single entity type, but many real-world systems involve multiple types of interacting entities with probabilistic dependencies that cross the boundaries of entities. The model in Figure 2(a) consists of three such dependencies (e.g., the competence of employees affect the success of products they develop). Many researchers have focused on representing domains characterized by these relational properties. In this section, we formally define the concepts of relational data and models, providing the basis for the theoretical framework for relational  $d$ -separation. We also show that the relational representation is strictly more expressive than the propositional representation used in Bayesian network modeling.

A relational schema is a top-level description of what data exist in a particular domain. Specifically (adapted from Heckerman et al., 2007):

**Definition 4.1 (Relational schema)** A *relational schema*  $\mathcal{S} = (\mathcal{E}, \mathcal{R}, \mathcal{A})$  consists of a set of entity classes  $\mathcal{E} = \{E_1, \dots, E_m\}$ ; relationship classes  $\mathcal{R} = \{R_1, \dots, R_n\}$ , where each  $R_i = \{E_1, \dots, E_j\}$  with  $E_j \in \mathcal{E}$ ; attribute classes  $\mathcal{A}(I)$  for each item  $I \in \mathcal{E} \cup \mathcal{R}$ ; and cardinality function  $card(R, E) = \{\text{ONE}, \text{MANY}\}$  for each  $R \in \mathcal{R}$  and each  $E \in \mathcal{E}$ .

A relational schema can be represented graphically with an entity-relationship (ER) diagram. We adopt a slightly modified ER diagram using Barker’s notation (1990), under which entities are rectangular boxes, relationships are diamonds with dashed lines connecting its associated entities, attributes are ovals residing on entities and relationships, and cardinalities are represented with crow’s foot notation.

**Example 4.1** The relational schema  $\mathcal{S}$  for the organization domain example depicted in Figure 1(a) consists of entities  $\mathcal{E} = \{\text{EMPLOYEE}, \text{PRODUCT}, \text{BUSINESS-UNIT}\}$ ; relationships  $\mathcal{R} = \{\text{DEVELOPS}, \text{FUNDS}\}$ , where  $\text{DEVELOPS} = \{\text{EMPLOYEE}, \text{PRODUCT}\}$ ,  $\text{FUNDS} = \{\text{BUSINESS-UNIT}, \text{PRODUCT}\}$  and having cardinalities  $card(\text{DEVELOPS}, \text{EMPLOYEE}) = \text{MANY}$ ,  $card(\text{DEVELOPS}, \text{PRODUCT}) = \text{MANY}$ ,  $card(\text{FUNDS}, \text{BUSINESS-UNIT}) = \text{ONE}$ , and  $card(\text{FUNDS}, \text{PRODUCT}) = \text{MANY}$ ; and attributes  $\mathcal{A}(\text{EMPLOYEE}) = \{\text{Competence}, \text{Salary}\}$ ,  $\mathcal{A}(\text{PRODUCT}) = \{\text{Success}\}$ , and  $\mathcal{A}(\text{BUSINESS-UNIT}) = \{\text{Budget}, \text{Revenue}\}$ .  $\square$

Propositional representations describe domains with a single entity class; thus, they produce simple schemas with  $|\mathcal{E}| = 1$  (one entity class) and  $|\mathcal{R}| = 0$  (no relationship classes). For the organization domain example, consider data about only employees ( $\mathcal{E} = \{\text{EMPLOYEE}\}$ ). Variables would include intrinsic attributes, such as competence and salary, but could also include variables describing other related entities, all from the employee perspective (see Figure 4(a) for the ER diagram of such a simple schema).<sup>3</sup> That is, we could construct a single table for employees that includes columns for the proportion of successfully developed products, the total revenue of all business units they work under, etc. Note that

3. This technique of translating a relational database down to a single, propositional representation is often referred to as *propositionalization* (Kramer et al., 2001).

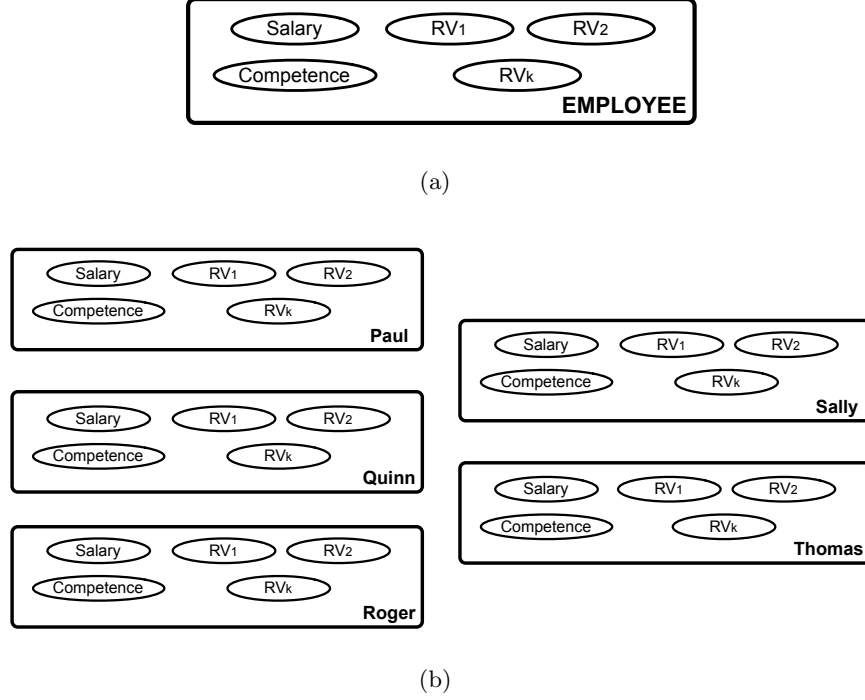


Figure 4: Relational schema (a) and skeleton (b) for a propositional representation of employees from the organization domain example. The skeleton consists of individual, disconnected entity instances. Each  $RV_i$  is a relational variable from the EMPLOYEE perspective, such as the proportion of products developed successfully or the total revenue of involved business units.

propositionalizing inherently relational data still requires the IID assumption, which is often violated since variables of one instance can influence variables of another. For example, according to the model in Figure 2(a) the competence of collaborating employees influences the success of products, which affects the revenue of business units, which affects its budget, thereby influencing an employee’s salary. As a result, modeling relational data with a propositional representation may unnecessarily lose valuable information, especially in the context of causal reasoning and accurate estimation of causal effects.

A relational schema is a template for the underlying relational skeleton (sometimes referred to as the data graph), a specific instantiation of entities and relationships. Specifically (adapted from Heckerman et al., 2007):

**Definition 4.2 (Relational skeleton)** A *relational skeleton*  $\sigma$  is an instantiation of entity sets  $\sigma(E)$  for each  $E \in \mathcal{E}$  and relationship sets  $\sigma(R)$  for each  $R \in \mathcal{R}$ , adhering to its cardinality constraints. Let  $r \in \sigma(R)$  where  $R = \{E_1, \dots, E_j\}$  be denoted as  $r(e_1, \dots, e_j)$  where  $e_i \in \sigma(E_i)$  and  $E_i \in \mathcal{E}$ .

**Example 4.2** The relational skeleton  $\sigma$  for the organization example is depicted in Figure 1(b). In this small skeleton, there are entity sets  $\sigma(\text{EMPLOYEE}) = \{\text{Paul, Quinn,}$

Roger, Sally, Thomas},  $\sigma(\text{PRODUCT}) = \{\text{Case, Adapter, Laptop, Tablet, Smartphone}\}$ , and  $\sigma(\text{BUSINESS-UNIT}) = \{\text{Accessories, Devices}\}$ . There are also relationship sets  $\sigma(\text{DEVELOPS}) = \{\text{develops}(\text{Paul, Case}), \text{develops}(\text{Quinn, Case}), \dots, \text{develops}(\text{Thomas, Smartphone})\}$  and  $\sigma(\text{FUNDS}) = \{\text{funds}(\text{Accessories, Case}), \text{funds}(\text{Accessories, Adapter}), \dots, \text{funds}(\text{Devices, Smartphone})\}$ . The relationship sets adhere to their cardinality constraints (e.g., FUNDS is a ONE-to-MANY relationship—within  $\sigma(\text{FUNDS})$ , every product has a single business unit, and every business unit may have multiple products).  $\square$

The relational skeleton of a propositional representation consists of a set of disconnected entity instances, all drawn from the same entity class. Consequently, the skeleton has a simple one-to-one mapping with the representation as a table: Each entity instance corresponds to a single row, and each variable is a column. Figure 4(b) shows a skeleton for the organization domain example had it been propositionalized from the employee perspective. Each employee is an entity instance with variables drawn from intrinsic attributes and possibly variables describing other related entities. No instances of other entity types or relationships exist in the skeleton of a propositional representation.

In order to specify a model over a *relational* domain, we must define a space of possible variables and dependencies. Consider the example dependency  $[\text{PRODUCT, DEVELOPS, EMPLOYEE}].\text{Competence} \rightarrow [\text{PRODUCT}].\text{Success}$  from the model in Figure 2(a). The variables here include an intrinsic entity attribute (success of a product) and also a variable on another entity (the competence of employees that develop a product). For relational data, the variable space includes not only intrinsic entity and relationship attributes, but also the variables on other entities and relationships that are reachable by paths along the relational skeleton. As above, paths in a relational skeleton are instantiations of path templates on a relational schema.

**Definition 4.3 (Relational path)** A *relational path*  $[I_1, \dots, I_k]$  for relational schema  $\mathcal{S}$  is an alternating sequence of entity and relationship classes  $I_1, \dots, I_k \in \mathcal{E} \cup \mathcal{R}$  such that for all  $j > 1$ : (1) if  $I_j \in \mathcal{E}$ , then  $I_{j-1} \in \mathcal{R}$  and  $I_j$  participates in  $I_{j-1}$  ( $I_j \in I_{j-1}$ ), (2) if  $I_j \in \mathcal{R}$ , then  $I_{j-1} \in \mathcal{E}$  and  $I_{j-1}$  participates in  $I_j$  ( $I_{j-1} \in I_j$ ), and (3) for each ordered triple  $\langle I_{j-1}, I_j, I_{j+1} \rangle$  in  $[I_1, \dots, I_k]$ , if  $I_j \in \mathcal{R}$ , then  $I_{j-1} \neq I_{j+1}$  and if  $I_j \in \mathcal{E}$ , then either  $I_{j-1} \neq I_{j+1}$  or  $\exists I_e \in I_{j-1}$  such that  $I_j \neq I_e$  and  $\text{card}(I_{j-1}, I_e) = \text{MANY}$ .  $I_1$  is called the *base item*, or *perspective*, of the relational path.

Definition 4.3 generalizes the notion of “slot chains” from the PRM framework (Getoor et al., 2007) by including cardinality constraints and formally describing the semantics under which repeated item classes may appear on a path. Conditions (2) and (3) in the definition essentially remove any paths that would invariably reach an empty terminal set (see Definition 4.4 below).<sup>4</sup> Also, since relational paths may become arbitrarily long, the path length is ordinarily limited by a user-specified, domain-specific hop threshold.

4. The condition  $I_j \in \mathcal{R} \Rightarrow I_{j-1} \neq I_{j+1}$  suggests at first glance that self-relationships (e.g., employees manage other employees, individuals in social networks maintain friendships, scholarly articles cite other articles) are prohibited. However, a common procedure in ER modeling is to map entity names to role names within the self-relationship (e.g., manager/subordinate, friend1/friend2, citing-paper/cited-paper), which accords with our definition of relational paths. For simplicity, we omit this additional layer of complexity.

**Example 4.3** From the example relational schema in Figure 1(a), we can derive the set of all relational paths up to some hop threshold. Let the hop threshold be  $h = 4$ . Then, the set of relational paths from the EMPLOYEE perspective would include the following: [EMPLOYEE] (employees, 0 hops), [EMPLOYEE, DEVELOPS, PRODUCT] (products developed by employees, 2 hops), [EMPLOYEE, DEVELOPS, PRODUCT, FUNDS, BUSINESS-UNIT] (business units of the products developed by employees, 4 hops), and [EMPLOYEE, DEVELOPS, PRODUCT, DEVELOPS, EMPLOYEE] (other employees developing the same products, 4 hops). Invalid relational paths would include [EMPLOYEE, DEVELOPS, EMPLOYEE] (because EMPLOYEE=EMPLOYEE and DEVELOPS $\in \mathcal{R}$ ) and [BUSINESS-UNIT, FUNDS, PRODUCT, FUNDS, BUSINESS-UNIT] (because PRODUCT $\in \mathcal{E}$  and  $\text{card}(\text{FUNDS, BUSINESS-UNIT}) = \text{ONE}$ ).  $\square$

An instantiated relational path produces a set of traversals on a relational skeleton. However, the quantity of interest is not the paths themselves, but the set of reachable items (i.e., entity or relationship instances):

**Definition 4.4 (Terminal set of a relational path)** For any skeleton  $\sigma$  and any  $i_1 \in \sigma(I_1)$ , a *terminal set*  $P|_{i_1}$  for relational path  $P = [I_1, \dots, I_k]$  can be defined inductively as

$$[I_1]|_{i_1} = \{i_1\}$$

$$[I_1, \dots, I_{k-1}, I_k]|_{i_1} = \bigcup_{i_{k-1} \in [I_1, \dots, I_{k-1}]|_{i_1}} \{i_k \mid ((i_{k-1} \in i_k \text{ if } I_k \in \mathcal{R}) \vee (i_k \in i_{k-1} \text{ if } I_k \in \mathcal{E})) \wedge i_k \notin [I_1, \dots, I_j]|_{i_1} \text{ for } j = 1 \text{ to } k-1\}$$

A terminal set of a relational path consists of reachable instances of class  $I_k$ , the terminal item on the path. Conceptually, a terminal set is produced by traversing the skeleton beginning at a single base item  $i_1 \in \sigma(I_1)$ , following instances of the items in the relational path, and reaching a target set of  $I_k$  instances. The definition implies a “bridge burning” semantics under which no instantiated items are revisited.<sup>5</sup>

**Example 4.4** We can generate terminal sets of a relational path by pairing the set of relational paths produced by the schema in Figure 1(a) with the relational skeleton in Figure 1(b). Let Quinn be our base item instance. Then  $[\text{EMPLOYEE}]|_{\text{Quinn}} = \{\text{Quinn}\}$ ,  $[\text{EMPLOYEE, DEVELOPS, PRODUCT}]|_{\text{Quinn}} = \{\text{Case, Adapter, Laptop}\}$ ,  $[\text{EMPLOYEE, DEVELOPS, PRODUCT, FUNDS, BUSINESS-UNIT}]|_{\text{Quinn}} = \{\text{Accessories, Devices}\}$ , and  $[\text{EMPLOYEE, DEVELOPS, PRODUCT, DEVELOPS, EMPLOYEE}]|_{\text{Quinn}} = \{\text{Paul, Roger, Sally}\}$ . The bridge burning semantics enforces that Quinn is not also included in this last terminal set.  $\square$

Most relational paths start and end with different item classes. However, there are pairs of distinct relational paths that start and end with the same item classes. For these pairs, it is possible that their terminal sets, when originating at the same base item instance, will have items in common. The following lemma states that if two relational paths with the same base and target items diverge in the middle of the path, then for some relational

5. The bridge burning semantics yields terminal sets that are necessarily subsets of terminal sets which would otherwise be produced without bridge burning. Although this appears to be limiting, it actually enables a strictly more expressive class of relational models. See Appendix B for more details and an example.

skeleton, their terminal sets will have a non-empty intersection. This property is important to consider for relational  $d$ -separation, and this is the only form for which non-empty intersection can occur.

**Lemma 4.1** *For any schema  $\mathcal{S}$  and any two relational paths  $P_1 = [I_1, \dots, I_m, \dots, I_k]$  and  $P_2 = [I_1, \dots, I_n, \dots, I_k]$  with  $I_m \neq I_n$ , there exists a skeleton  $\sigma$  such that  $P_1|_{i_1} \cap P_2|_{i_1} \neq \emptyset$  for some  $i_1 \in \sigma(I_1)$ .*

**Proof.** See Appendix A.

**Example 4.5** Let  $\text{Path}_1 = [\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}]$ , the terminal sets for which yields other products developed by collaborating employees. Let  $\text{Path}_2 = [\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{FUNDS}, \text{BUSINESS-UNIT}, \text{FUNDS}, \text{PRODUCT}]$ , for which terminal sets consist of other products funded by the business units funding products developed by a given employee. Intersection among terminal sets for these paths occurs even in this small skeleton. In fact, the intersection of the terminal sets for  $\text{Path}_1$  and  $\text{Path}_2$  is non-empty for all employees. For example, Paul:  $\text{Path}_1|_{\text{Paul}} = \{\text{Adapter}, \text{Laptop}\}$  and  $\text{Path}_2|_{\text{Paul}} = \{\text{Adapter}\}$ ; Quinn:  $\text{Path}_1|_{\text{Quinn}} = \{\text{Tablet}\}$  and  $\text{Path}_2|_{\text{Quinn}} = \{\text{Tablet}, \text{Smartphone}\}$ .  $\square$

Given the definition for relational paths, it is simple to define relational variables and their terminal sets.

**Definition 4.5 (Relational variable)** A *relational variable*  $[I_1, \dots, I_k].V$  for schema  $\mathcal{S}$  consists of a relational path  $[I_1, \dots, I_k]$  and an attribute class  $V \in \mathcal{A}(I_k)$ .

**Example 4.6** Relational variables for the relational paths in Example 4.3 include intrinsic variables such as  $[\text{EMPLOYEE}].\text{Competence}$  and  $[\text{EMPLOYEE}].\text{Salary}$ , and also variables on related entities such as  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}].\text{Success}$ ,  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{FUNDS}, \text{BUSINESS-UNIT}].\text{Revenue}$ , and  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}].\text{Salary}$ .  $\square$

**Definition 4.6 (Terminal set of a relational variable)** For any skeleton  $\sigma$  and  $i_1 \in \sigma(I_1)$ , a *terminal set*  $P.V|_{i_1}$  for relational variable  $P.V = [I_1, \dots, I_k].V$  is the set of variable instances  $\{i_k.V \mid V \in \mathcal{A}(i_k) \wedge i_k \in P|_{i_1}\}$ .

**Example 4.7** Terminal sets for base item instance Sally and the relational variables from Example 4.6 include  $[\text{EMPLOYEE}].\text{Competence}|_{\text{Sally}} = \{\text{Sally}.\text{Competence}\}$ ,  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}].\text{Success}|_{\text{Sally}} = \{\text{Laptop}.\text{Success}, \text{Tablet}.\text{Success}\}$ ,  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{FUNDS}, \text{BUSINESS-UNIT}].\text{Revenue}|_{\text{Sally}} = \{\text{Devices}.\text{Revenue}\}$ , and  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}].\text{Salary}|_{\text{Sally}} = \{\text{Quinn}.\text{Salary}, \text{Thomas}.\text{Salary}\}$ .  $\square$

As a notational convenience, if  $\mathbf{X}$  is a set of relational variables, all from a common perspective  $I_1$ , then we say that  $\mathbf{X}|_{i_1}$  for some item  $i_1 \in \sigma(I_1)$  is the union of all terminal sets,  $\{x \mid x \in X|_{i_1} \wedge X \in \mathbf{X}\}$ . Given the formal definitions for relational variables, we can now define relational dependencies.

**Definition 4.7 (Relational dependency)** A *relational dependency*  $[I_1, \dots, I_k].V_1 \rightarrow [I_1].V_2$  consists of two relational variables with a common base item and corresponds to a directed probabilistic dependence from  $[I_1, \dots, I_k].V_1$  to  $[I_1].V_2$ .

Depending on the context,  $[I_1, \dots, I_k].V_1$  and  $[I_1].V_2$  can be referred to as *treatment* and *outcome*, *cause* and *effect*, or *parent* and *child*. Without loss of generality, Definition 4.7 provides a canonical specification for dependencies, with the child relational variable restricted to singleton paths, thus ensuring that the terminal sets of a child relational variable consist of a single value.

**Example 4.8** The dependencies in the relational model displayed in Figure 2(a) can be specified as:  $[\text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}].\text{Competence} \rightarrow [\text{PRODUCT}].\text{Success}$  (product success is influenced by the competence of the employees developing the product),  $[\text{EMPLOYEE}].\text{Competence} \rightarrow [\text{EMPLOYEE}].\text{Salary}$  (an employee’s competence affects her salary),  $[\text{BUSINESS-UNIT}, \text{FUNDS}, \text{PRODUCT}].\text{Success} \rightarrow [\text{BUSINESS-UNIT}].\text{Revenue}$  (the success of the products funded by a business unit influences that unit’s revenue),  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{FUNDS}, \text{BUSINESS-UNIT}].\text{Budget} \rightarrow [\text{EMPLOYEE}].\text{Salary}$  (employee salary is governed by the budget of the business units for which they develop products), and  $[\text{BUSINESS-UNIT}].\text{Revenue} \rightarrow [\text{BUSINESS-UNIT}].\text{Budget}$  (the revenue of a business unit influences its budget).  $\square$

A relational model is simply a collection of relational dependencies, defined as:

**Definition 4.8 (Relational model)** The structure of a *relational model*  $\mathcal{M} = (\mathcal{S}, \mathcal{D})$  consists of a schema  $\mathcal{S}$  paired with a set of relational dependencies  $\mathcal{D}$  defined over  $\mathcal{S}$ .

A relational model can be represented graphically by superimposing dependencies on the ER diagram of a relational schema (see Figure 2(a) for an example). This definition of relational models is consistent with and yields structures expressible as DAPER models (Heckerman et al., 2007). These relational models are also equivalent to PRMs, but we generalize slot chains as relational paths and provide a formal semantics for their instantiation (i.e., terminal sets of relational paths and relational variables). These models are also more general than plate models because dependencies can be specified with arbitrary relational paths as opposed to simple intersections among plates (Buntine, 1994; Gilks et al., 1994).

Relational models only define coherent joint probability distributions if they produce acyclic model instantiations (i.e., ground graphs, defined below). A useful construct for checking model acyclicity is the class dependency graph (Getoor et al., 2007), defined as:

**Definition 4.9 (Class dependency graph)** The class dependency graph  $G_{\mathcal{M}} = (V, E)$  for relational model  $\mathcal{M} = (\mathcal{S}, \mathcal{D})$  is a directed graph with nodes for each attribute of each item class  $V = \{I.V \mid I \in \mathcal{E} \cup \mathcal{R} \wedge V \in \mathcal{A}(I)\}$  and edges between pairs of attributes supported by relational dependencies in the model  $E = \{I_k.V_1 \rightarrow I_j.V_2 \mid [I_j, \dots, I_k].V_1 \rightarrow [I_j].V_2 \in \mathcal{D}\}$ .

If the relational dependencies form an acyclic class dependency graph, then every possible ground graph of that model is acyclic as well (Getoor et al., 2007).<sup>6</sup> All future references

6. Getoor et al. (2007) additionally define *guaranteed acyclic relationships* which enable cyclic relational dependencies that are guaranteed to produce only acyclic instantiations. The class dependency graphs are accordingly “colored,” and acyclicity is checked with respect to stratification across colors. We omit this type of relationship for simplicity of developing the relational  $d$ -separation theory.

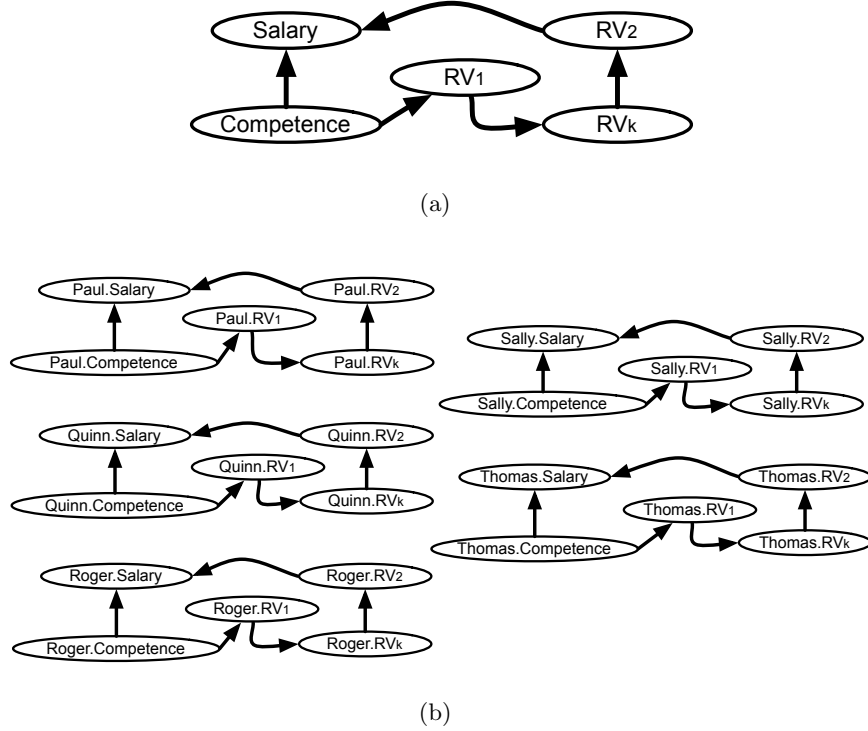


Figure 5: Example Bayesian network (a) and ground graph (b) for the propositional representation of employees from the organization domain. The model consists of simple dependencies involving employee variables. The ground graph is a set of identical copies of the model structure, one for each instance in the data (skeleton).

to acyclic relational models refer to relational models whose dependency structure forms acyclic class dependency graphs. A parameterized relational model contains conditional probability distributions for every attribute class  $\mathcal{A}(I)$  for each  $I \in \mathcal{E} \cup \mathcal{R}$  in order to represent a joint probability distribution. Similar to Bayesian networks, the joint distribution factorizes according to the conditional distributions given a relational skeleton  $\sigma$  as

$$P(GG_{\mathcal{M}\sigma}) = \prod_{I \in \mathcal{E} \cup \mathcal{R}} \prod_{X \in \mathcal{A}(I)} \prod_{i \in \sigma(I)} P(i.X \mid \text{parents}(i.X))$$

where  $GG_{\mathcal{M}\sigma}$  is the ground graph of the model and skeleton, defined below. Note that without a generative model of relational skeletons, these relational models are not truly generative as the skeleton must be generated prior to the attributes. However, the same issue occurs for Bayesian networks—relational skeletons consist of disconnected entity instances, but the number of such instances to create is not described by the model. We choose simple processes for generating skeletons, allowing us to focus on relational models of attributes and leaving structural causes and effects as future work.

A common propositional model is the Bayesian network, as described in Section 3. Because all variables are defined for a single entity type and no relationships exist, the relational path specification becomes trivial and, hence, implicit. All relational paths, relational variables, and relational dependencies are defined from a single perspective with singleton paths (e.g., [EMPLOYEE]). Figure 5(a) depicts an example Bayesian network for the schema in Figure 4(a). The entity box surrounding the model is implicit given a single entity type.

Just as the relational schema is a template for skeletons, a relational model can be viewed as a template for ground graphs—how dependencies apply to skeletons.

**Definition 4.10 (Ground graph)** The *ground graph*  $GG_{\mathcal{M}\sigma} = (V, E)$  for relational model  $\mathcal{M} = (\mathcal{S}, \mathcal{D})$  and skeleton  $\sigma$  is a directed graph with nodes  $V = \mathcal{A}(\sigma) = \{i.X \mid I \in \mathcal{E} \cup \mathcal{R} \wedge X \in \mathcal{A}(I) \wedge i \in \sigma(I)\}$  and edges  $E = \{i_k.Y \rightarrow i_j.X \mid i_k.Y, i_j.X \in V \wedge i_k.Y \in [I_j, \dots, I_k].Y|_{i_j} \wedge [I_j, \dots, I_k].Y \rightarrow [I_j].X \in \mathcal{D}\}$ .

A ground graph is a large directed graph, with a node for every variable of every entity and relationship instance in a skeleton, and an edge between pairs of variable instances that belong to the terminal sets of the relational variables of all dependencies in a model. In fact, given an acyclic relational model, the ground graph has the same semantics as a Bayesian network (Getoor, 2001; Heckerman et al., 2007). The ground graph in Figure 2(b) is the result of applying the dependencies in the relational model shown in Figure 2(a) to the skeleton in Figure 1(b).

The ground graph of a Bayesian network, like the skeleton of a propositional schema, has a very regular structure. The ground graph consists of a set of independent identical instances of the model structure, one for each instance in the data. Figure 5(b) shows the resulting ground graph of applying the model to the data, or skeleton, of Figure 4(b).

By Lemma 4.1 and Definition 4.10, it is clear that the same canonical dependency connects many other relational variables. If the terminal sets involve  $i_k.Y$  and  $i_j.X$ , then there is an implied dependency between all other relational variables for which  $i_k.Y$  and  $i_j.X$  are elements.

**Example 4.9** The canonical dependency  $[\text{EMPLOYEE}].\text{Competence} \rightarrow [\text{EMPLOYEE}].\text{Salary}$  yields the edge  $\text{Roger}.\text{Competence} \rightarrow \text{Roger}.\text{Salary}$  in the ground graph of Figure 2(b) because  $\text{Roger}.\text{Competence} \in [\text{EMPLOYEE}].\text{Competence}|_{\text{Roger}}$ . However,  $\text{Roger}.\text{Competence} \in [\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}].\text{Competence}|_{\text{Salary}}$  (as is  $\text{Roger}.\text{Salary}$ , replacing *Competence* with *Salary*). Consequently, the canonical dependency *implies* dependence among the variable instances in the terminal sets for  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}].\text{Competence}$  and  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}].\text{Salary}$ .  $\square$

These implied dependencies form the crux of the challenge of identifying independence in relational models. Additionally, the intersection between the terminal sets of two relational paths is crucial for reasoning about independence because variable instances can belong to the terminal sets of multiple relational variables. Since *d*-separation only guarantees independence when there are no *d*-connecting paths, we must consider all possible paths among variable instances, any one of which may be a member of multiple relational variables.



In Section 5, we define relational  $d$ -separation and provide an appropriate representation, the abstract ground graph, that enables straightforward reasoning about  $d$ -separation.

## 5. Relational $d$ -separation

Conditional independence facts are entailed by the rules of  $d$ -separation, but only for simple directed acyclic graphs. As described in Section 4, every ground graph of a Bayesian network consists of a set of identical copies of the model structure. Thus, the implications of  $d$ -separation on Bayesian networks hold for every ground graph. In contrast, relational models are templates for ground graphs that vary with underlying skeletons. That is, the set of distributions represented by a relational model is not only parameterized by conditional probabilities, but also by the space of valid skeletons given by the schema. Since conditional independence facts are only useful when they hold across all possible model instantiations, reasoning about  $d$ -separation for relational models is inherently more challenging and leads to the following definition:

**Definition 5.1 (Relational  $d$ -separation)** Let  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $\mathbf{Z}$  be three distinct sets of relational variables for perspective  $B \in \mathcal{E} \cup \mathcal{R}$  defined over relational schema  $\mathcal{S}$ . Then, for relational model  $\mathcal{M}$ ,  $\mathbf{X}$  and  $\mathbf{Y}$  are  $d$ -separated by  $\mathbf{Z}$  if and only if, for any skeleton  $\sigma$ ,  $\mathbf{X}|_b$  and  $\mathbf{Y}|_b$  are  $d$ -separated by  $\mathbf{Z}|_b$  in ground graph  $GG_{\mathcal{M}\sigma}$  for all  $b \in \sigma(B)$ .

In other words, for  $\mathbf{X}$  and  $\mathbf{Y}$  to be  $d$ -separated by  $\mathbf{Z}$  for relational model  $\mathcal{M}$ ,  $d$ -separation must hold for all instantiations of those relational variables for any possible skeleton. This is a conservative definition, but it is consistent with the semantics of  $d$ -separation on Bayesian networks—it guarantees independence, but it does not guarantee dependence. If there exists even one valid skeleton and faithful distribution represented by the relational model for which  $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ , then  $\mathbf{X}$  and  $\mathbf{Y}$  are not  $d$ -separated by  $\mathbf{Z}$ .

Given the semantics specified in Definition 5.1, answering relational  $d$ -separation queries is challenging for the following reasons:

*All-ground-graphs semantics:* Although it is possible to verify  $d$ -separation on a single ground graph of a relational model, the conclusion may not generalize and ground graphs can be arbitrarily large. The semantics of  $d$ -separation on Bayesian networks also holds for all possible ground graphs. However, the ground graphs of a Bayesian network consist of identical copies of the structure of the model, and it is sufficient to reason about  $d$ -separation on a single subgraph.

*Relational models are templates:* Relational models may appear to be directed acyclic graphs, but they are templates for constructing ground graphs. The rules of  $d$ -separation do not directly apply to relational models, only to its ground graphs. Applying the rules of  $d$ -separation to a relational model frequently leads to incorrect conclusions because of unrepresented confounding paths that are only manifest in ground graphs.

*Terminal sets of relational variables may intersect:* The terminal sets of two different relational variables may have non-empty intersections, as described by Lemma 4.1. Consequently, there exist non-intuitive implications of dependencies that  $d$ -separation must account for, such as the relational  $d$ -connecting paths in the example in Section 2.

*Relational dependency specification:* Relational models are defined with canonical dependencies, each specified from a single perspective. However, variables in a ground graph

may contribute to the terminal sets of *multiple* relational variables, each defined from *different* perspectives. Thus, we need methods to translate and extend canonically specified dependencies to produce the implied dependencies between arbitrary relational variables, such as the implied dependency described in Example 4.9.

### 5.1 Abstracting over All Ground Graphs

The definition of relational  $d$ -separation and its challenges suggest a solution that abstracts over all possible ground graphs and explicitly represents the potential intersection between the terminal sets of pairs of relational variables. We introduce a new lifted representation, called the *abstract ground graph*, that captures all dependencies among arbitrary relational variables for any ground graph, using knowledge of only the schema and the model.

**Definition 5.2 (Abstract ground graph)** An *abstract ground graph*  $AGG_{\mathcal{M}Bh} = (V, E)$  for relational model  $\mathcal{M} = (\mathcal{S}, \mathcal{D})$ , perspective  $B \in \mathcal{E} \cup \mathcal{R}$ , and hop threshold  $h \in \mathbb{N}^0$  is an abstraction of the dependencies  $\mathcal{D}$  for all possible ground graphs  $GG_{\mathcal{M}\sigma}$  of  $\mathcal{M}$  on arbitrary skeletons  $\sigma$ .

The set of nodes in  $AGG_{\mathcal{M}Bh}$ ,  $V = RV \cup IV$ , is the union of all relational variables  $RV = \{[B, \dots, I_j].V \mid \text{length}([B, \dots, I_j]) \leq h+1\}$  and the intersections between pairs of relational variables that could intersect  $IV = \{X \cap Y \mid X, Y \in RV \wedge X = [B, \dots, I_k, \dots, I_j].V \wedge Y = [B, \dots, I_l, \dots, I_j].V \wedge I_k \neq I_l\}$ .

The set of edges in  $AGG_{\mathcal{M}Bh}$  is  $E = RVE \cup IVE$ , where  $RVE \subset RV \times RV$  and  $IVE \subset IV \times RV \cup RV \times IV$ .  $RVE$  is the set of edges between pairs of relational variables:  $RVE = \{[B, \dots, I_k].V_1 \rightarrow [B, \dots, I_j].V_2 \mid [I_j, \dots, I_k].V_1 \rightarrow [I_j].V_2 \in \mathcal{D} \wedge [B, \dots, I_k] \in \text{extend}([B, \dots, I_j], [I_j, \dots, I_k])\}$ .

$IVE$  is the set of edges inherited from both relational variable sources of every intersection variable:  $IVE = \{X \rightarrow [B, \dots, I_j].V_2 \mid X = P_1.V_1 \cap P_2.V_1 \in IV \wedge (P_1.V_1 \rightarrow [B, \dots, I_j].V_2 \in RVE \vee P_2.V_1 \rightarrow [B, \dots, I_j].V_2 \in RVE)\} \cup \{[B, \dots, I_j].V_1 \rightarrow X \mid X = P_1.V_2 \cap P_2.V_2 \in IV \wedge ([B, \dots, I_j].V_1 \rightarrow P_1.V_1 \in RVE \vee [B, \dots, I_j].V_1 \rightarrow P_2.V_1 \in RVE)\}$ .

The *extend* method is described in Definition 5.3 below. Essentially, the construction of an abstract ground graph for relational model  $\mathcal{M}$ , perspective  $B \in \mathcal{E} \cup \mathcal{R}$ , and hop threshold  $h$  follows three simple steps: (1) Add a node for all relational variables, with relational path length limited by  $h$ . (2) Insert edges for the direct causes of every relational variable. (3) For each pair of potentially intersecting relational variables, add a new “intersection” node that inherits the direct causes and effects from both of its sources. Then, to answer queries of the form “Are  $\mathbf{X}$  and  $\mathbf{Y}$   $d$ -separated by  $\mathbf{Z}$ ?”, simply (1) augment  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $\mathbf{Z}$  with their corresponding intersection variables and (2) apply the rules of  $d$ -separation on the abstract ground graph for the common perspective of  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $\mathbf{Z}$ .

**Example 5.1** Figure 6 shows the abstract ground graph  $AGG_{\mathcal{M}\text{EMPLOYEE}6}$  for the organization example from the EMPLOYEE perspective with hop threshold  $h = 6$ .<sup>7</sup> As in Section 2, we derive an appropriate conditioning set  $\mathbf{Z}$  in order to  $d$ -separate individual employee competence ( $\mathbf{X} = \{[\text{EMPLOYEE}].\text{Competence}\}$ ) from the revenue of the employee’s funding business units ( $\mathbf{Y} = \{[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{FUNDS}, \text{BUSINESS-UNIT}].\text{Revenue}\}$ ).

7. The variables *Salary* and *Budget* are removed for simplicity. They are irrelevant for this  $d$ -separation example as they are solely effects of other variables.

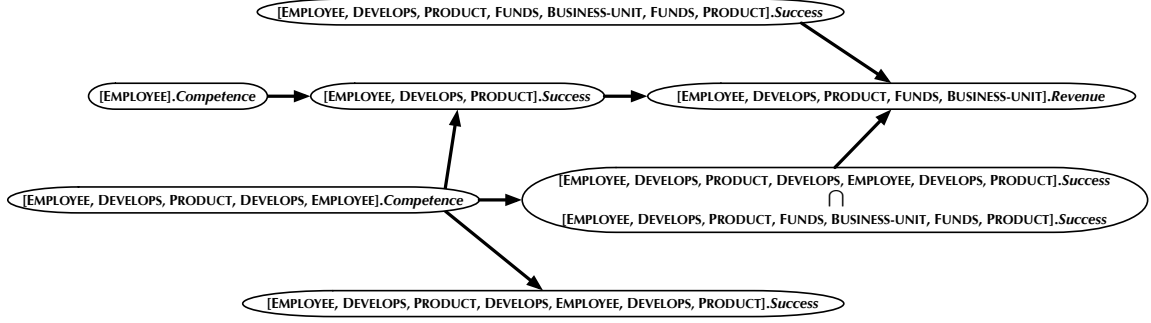


Figure 6: The abstract ground graph for the organization domain model in Figure 2(a) from the EMPLOYEE perspective with hop threshold  $h = 6$  (with the variables for *Salary* and *Budget* omitted for simplicity). This abstract ground graph includes one intersection node.

Applying the rules of  $d$ -separation to the abstract ground graph, we see that it is necessary to condition on both product success ( $[EMPLOYEE, DEVELOPS, PRODUCT].Success$ ) and the competence of other employees developing the same products ( $[EMPLOYEE, DEVELOPS, PRODUCT, DEVELOPS, EMPLOYEE].Competence$ ). For  $h = 6$ , augmenting  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $\mathbf{Z}$  with their corresponding intersection variables does not result in any changes. For  $h = 8$ , the abstract ground graph includes a node for relational variable  $[EMPLOYEE, DEVELOPS, PRODUCT, DEVELOPS, EMPLOYEE, DEVELOPS, PRODUCT].Success$  (the revenue of the business units funding the other products of collaborating employees) which, by Lemma 4.1, could have a non-empty intersection with  $[EMPLOYEE, DEVELOPS, PRODUCT, FUNDS, BUSINESS-UNIT].Revenue$ . Therefore,  $\mathbf{Y}$  would also include the intersection with this other relational variable. However, for this query, the conditioning set  $\mathbf{Z}$  for  $h = 6$  happens to also  $d$ -separate at  $h = 8$  (and any  $h \in \mathbb{N}^0$ ).  $\square$

Using the algorithm devised by Geiger et al. (1990), relational  $d$ -separation queries can be answered in  $O(|E|)$  time with respect to the number of edges in the abstract ground graph. In practice, the size of an abstract ground graph depends on the relational schema and model (i.e., the number of entities, relationships, cardinalities, attributes, and dependencies—see the experiment in Section 7.1), as well as the hop threshold limiting the length of relational paths. For the example in Figure 6, the abstract ground graph has 7 nodes and 7 edges (including 1 intersection node with 2 edges); for  $h = 8$ , it would have 13 nodes and 21 edges (including 4 intersection nodes with 13 edges). Abstract ground graphs are invariant to the size of ground graphs, even though ground graphs can be arbitrarily large—that is, relational databases have no maximum size.

Next, we formally define the *extend* method, used internally for the construction of abstract ground graphs. This method translates dependencies specified in the model into dependencies in the abstract ground graph.

**Definition 5.3 (Extending relational paths)** Let  $P_{orig} = [I_1, \dots, I_j]$  and let  $P_{ext} = [I_j, \dots, I_k]$  be two relational paths for schema  $\mathcal{S}$ . The following three functions extend  $P_{orig}$

with  $P_{ext}$ :

$$extend(P_{orig}, P_{ext}) = \{P = concat(P_{orig}[0 : length(P_{orig}) - i + 1], P_{ext}[i : length(P_{ext})]) \mid i \in pivots(reverse(P_{orig}), P_{ext}) \wedge isValid(P)\}$$

$$pivots(P_1, P_2) = \{i \mid P_1[0 : i] = P_2[0 : i]\}$$

$$isValid(P) = \begin{cases} \text{True} & \text{if } P \text{ does not violate Definition 4.3} \\ \text{False} & \text{otherwise} \end{cases}$$

where *concat*, *length*, *reverse*, and  $[i:j]$  inclusive-exclusive sublist are standard list functions.

Informally, the *extend* method constructs a set of valid relational paths from two input relational paths. It first finds the indices of all items for which the input paths have a common starting subpath (called pivots), and then it concatenates the two paths at each pivot, removing one of the duplicated subpaths (see Example 5.2). Since *d*-separation requires blocking all paths of dependence between two sets of variables, the *extend* method is critical to ensure the soundness and completeness of our approach. The abstract ground graph must capture all paths of dependence among the variable instances in the terminal sets of relational variables for any represented ground graph. However, relational models are specified solely with respect to canonical dependencies; the *extend* method is called repeatedly during the creation of an abstract ground graph, with  $P_{orig}$  set to a given relational path and  $P_{ext}$  drawn from a canonical dependency.

**Example 5.2** During the construction of the abstract ground graph  $AGG_{\mathcal{M}_{\text{EMPLOYEE6}}}$  depicted in Figure 6, the *extend* method is called several times. First, all relational variables for  $h = 6$  from the EMPLOYEE perspective are added as nodes in the graph. Next, *extend* is used to insert edges corresponding to direct causes. Consider the node for [EMPLOYEE, DEVELOPS, PRODUCT].*Success*. The construction of  $AGG_{\mathcal{M}_{\text{EMPLOYEE6}}}$  calls  $extend(P_{orig}, P_{ext})$  with  $P_{orig} = [\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}]$  and  $P_{ext} = [\text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}]$  because  $[\text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}].\text{Competence} \rightarrow [\text{PRODUCT}].\text{Success} \in \mathcal{D}$ . Here,  $extend(P_{orig}, P_{ext}) = \{[\text{EMPLOYEE}], [\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}]\}$ , which leads to the insertion of two edges in the abstract ground graph. Note that  $pivots(reverse(P_{orig}), P_{ext}) = \{1, 2, 3\}$ , and the pivot at  $i = 2$  yields the invalid relational path [EMPLOYEE, DEVELOPS, EMPLOYEE].  $\square$

We also describe two important properties of the *extend* method with the following two lemmas. The first lemma states that every relational path produced by the *extend* method yields a terminal set for some skeleton such that there is an item instance also reachable by the two original paths. This lemma is useful for proving the soundness of our abstraction—that all edges inserted in an abstract ground graph correspond to edges in some ground graph.

**Lemma 5.1** *For any two relational paths  $P_{orig} = [I_1, \dots, I_j]$  and  $P_{ext} = [I_j, \dots, I_k]$  with  $\mathbf{P} = extend(P_{orig}, P_{ext})$ ,  $\forall P \in \mathbf{P}$  there exists a relational skeleton  $\sigma$  such that  $\exists i_1 \in \sigma(I_1)$  such that  $\exists i_k \in P|_{i_1}$  and  $\exists i_j \in P_{orig}|_{i_1}$  such that  $i_k \in P_{ext}|_{i_j}$ .*

**Proof.** See Appendix A.

**Example 5.3** Let  $\sigma$  be the skeleton shown in Figure 1(b), let  $P_{orig} = [\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}]$ , let  $P_{ext} = [\text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}]$ , and let  $i_1 = \text{Sally} \in \sigma(\text{EMPLOYEE})$ . From Example 5.2, we know that  $\mathbf{P} = \text{extend}(P_{orig}, P_{ext}) = \{[\text{EMPLOYEE}], [\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}]\}$ . We also have  $[\text{EMPLOYEE}]|_{\text{Sally}} = \{\text{Sally}\}$  and  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}]|_{\text{Sally}} = \{\text{Quinn}, \text{Roger}, \text{Thomas}\}$ . By Lemma 5.1, there should exist an  $i_j \in P_{orig}|_{i_1}$  for each of these employees (Quinn, Roger, Sally, and Thomas) such that they are in the terminal set  $P_{ext}|_{i_j}$ . We have  $P_{orig}|_{\text{Sally}} = \{\text{Laptop}, \text{Tablet}\}$ , and  $P_{ext}|_{\text{Laptop}} = \{\text{Quinn}, \text{Roger}, \text{Sally}\}$  and  $P_{ext}|_{\text{Tablet}} = \{\text{Sally}, \text{Thomas}\}$ . So, the lemma clearly holds for this example.  $\square$

Lemma 5.1 guarantees that, for some relational skeleton, items in the terminal sets produced by *extend* are also reachable in combination of the two input paths,  $P_{orig}$  and  $P_{ext}$ . It is also possible (although infrequent) that there exist items reachable by  $P_{orig}$  and  $P_{ext}$  that are not in the terminal set of any path produced with  $\text{extend}(P_{orig}, P_{ext})$ . The following lemma describes this unreachable set of items, stating that there must exist an alternative relational path  $P'_{orig}$  that intersects with  $P_{orig}$  that, when using *extend*, catches those remaining items. This lemma is important for proving the completeness of our abstraction—that all edges in any ground graph are represented in the abstract ground graph.

**Lemma 5.2** *For any relational skeleton  $\sigma$  and two relational paths  $P_{orig} = [I_1, \dots, I_j]$  and  $P_{ext} = [I_j, \dots, I_k]$  with  $\mathbf{P} = \text{extend}(P_{orig}, P_{ext})$ ,  $\forall i_1 \in \sigma(I_1) \forall i_j \in P_{orig}|_{i_1} \forall i_k \in P_{ext}|_{i_j}$  if  $\forall P \in \mathbf{P} i_k \notin P|_{i_1}$ , then  $\exists P'_{orig}$  such that  $P_{orig}|_{i_1} \cap P'_{orig}|_{i_1} \neq \emptyset$  and  $i_k \in P'|_{i_1}$  for some  $P' \in \text{extend}(P'_{orig}, P_{ext})$ .*

**Proof.** See Appendix A.

**Example 5.4** Although Lemma 5.2 does not apply to the organization domain as currently represented, it could apply if either (1) there were cycles in the relational schema or (2) if the path specifications on the relational dependencies included a cycle. Consider additional relationships between employees and products. If employees could be involved with products at various stages (e.g., research, development, testing, marketing), then there would be alternative relational paths for which the lemma might apply. The proof of the lemma in Appendix A provides abstract conditions describing when the lemma applies.  $\square$

## 5.2 Proof of Correctness

The correctness of our approach to relational  $d$ -separation relies on several facts: (1)  $d$ -separation is valid for directed acyclic graphs (DAGs); (2) ground graphs are DAGs; and (3) abstract ground graphs are directed acyclic graphs that represent exactly the edges that could appear in all possible ground graphs. It follows that  $d$ -separation on abstract ground graphs, augmented by intersection variables, is sound and complete for all ground graphs. Additionally, we show that since relational  $d$ -separation is sound and complete, it is also equivalent to the Markov condition for relational models. Using the previous definitions and lemmas, the following sequence of results proves the correctness of our approach to identifying independence in relational models.

**Theorem 5.1** *The rules of  $d$ -separation are sound and complete for directed acyclic graphs.*

**Proof.** Due to Verma and Pearl (1988) for soundness and Geiger and Pearl (1988) for completeness. ■

Theorem 5.1 implies that (1) all conditional independence facts derived by  $d$ -separation on a Bayesian network hold in any faithful distribution represented by that model (soundness) and (2) all conditional independence facts that hold in a faithful distribution can be inferred from  $d$ -separation applied to the Bayesian network encoding that distribution (completeness).

**Lemma 5.3** *For any acyclic relational model  $\mathcal{M}$  and skeleton  $\sigma$ , the ground graph  $GG_{\mathcal{M}\sigma}$  is a directed acyclic graph.*

**Proof.** Due to both Heckerman et al. (2007) for DAPER models and Getoor (2001) for PRMs. ■

Lemma 5.3 states that any ground graph of an acyclic relational model is essentially a very large Bayesian network. By Theorem 5.1,  $d$ -separation is sound and complete when applied directly to a ground graph. However, Definition 5.1 explicitly states that relational  $d$ -separation must hold across *all possible* ground graphs, which is the reason for constructing the abstract ground graph representation. Next, we introduce the notion of  $(B, h)$ -reachability, which describes the conditions under which we can expect an edge in a ground graph to be represented in an abstract ground graph.

**Definition 5.4** ( $(B, h)$ -reachability) Let  $GG_{\mathcal{M}\sigma}$  be the ground graph for some relational model  $\mathcal{M}$  and skeleton  $\sigma$ . Then,  $i_k.V_1 \rightarrow i_j.V_2 \in GG_{\mathcal{M}\sigma}$  is  $(B, h)$ -reachable for perspective  $B$  and hop threshold  $h$  if there exist relational variables  $P_k.V_1 = [B, \dots, I_k].V_1$  and  $P_j.V_2 = [B, \dots, I_j].V_2$  such that  $\text{length}(P_k) \leq h + 1$ ,  $\text{length}(P_j) \leq h + 1$ , and there exists a  $b \in \sigma(B)$  where  $i_k \in P_k|_b$  and  $i_j \in P_j|_b$ .

**Example 5.5** Consider the ground graph shown in Figure 2(b). Let perspective  $B$  be EMPLOYEE, and let the hop threshold  $h = 6$ . Let  $i_k.V_1 \rightarrow i_j.V_2$  be the edge Laptop.Success  $\rightarrow$  Devices.Revenue in the ground graph. Then, we can show that this edge is  $(B, h)$ -reachable because of the following: Set  $P_k.V_1 = [\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}].\text{Success}$ ,  $P_j.V_2 = [\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{FUNDS}, \text{BUSINESS-UNIT}].\text{Revenue}$ , and let  $b = \text{Sally} \in \sigma(\text{EMPLOYEE})$ . We have  $\text{length}(P_k) = 3 < 7$ ,  $\text{length}(P_j) = 5 < 7$ , Laptop  $\in P_k|_{\text{Sally}}$ , and Devices  $\in P_j|_{\text{Sally}}$ . □

Since Definition 5.4 pertains to edges reachable via a particular perspective  $B$  and hop threshold  $h$ , it relates to the reachability of edges in abstract ground graphs. Specifically, Definition 5.4 implies that (1) for any edge in ground graph  $GG_{\mathcal{M}\sigma}$ , we can derive a set of abstract ground graphs for which that edge is  $(B, h)$ -reachable, and (2) for any abstract ground graph  $AGG_{\mathcal{M}Bh}$ , we can derive the set of  $(B, h)$ -reachable edges for a given ground graph. Given  $(B, h)$ -reachability, we can now express the soundness and completeness of abstract ground graphs, which is depicted graphically in Figure 7.

$$AGG_{\mathcal{M}Bh} \xrightleftharpoons[(B,h)\text{-completeness}]{(B,h)\text{-soundness}} GG_{\mathcal{M}\sigma}$$

Figure 7: A relational model  $\mathcal{M}$  combined with a perspective  $B$  and a hop threshold  $h$  yields the abstract ground graph  $AGG_{\mathcal{M}Bh}$ . The model combines with a skeleton  $\sigma$  to produce the ground graph  $GG_{\mathcal{M}\sigma}$ . In Theorem 5.2, we show that abstract ground graphs are both sound and complete with respect to  $(B, h)$ -reachability.

**Theorem 5.2** *For any acyclic relational model  $\mathcal{M}$ , perspective  $B \in \mathcal{E} \cup \mathcal{R}$ , and hop threshold  $h \in \mathbb{N}^0$ , the abstract ground graph  $AGG_{\mathcal{M}Bh}$  is  $(B, h)$ -reachably sound and complete for any ground graph  $GG_{\mathcal{M}\sigma}$  for all skeletons  $\sigma$ .*

**Proof.** See Appendix A.

Theorem 5.2 guarantees that, up to the hop threshold for a given perspective, abstract ground graphs capture all possible paths of dependence between any pair of variables in any ground graph. The details of the proof provide the reasons why explicitly representing the intersection between pairs of relational variables is necessary for ensuring a sound and complete abstraction.

**Theorem 5.3** *For any acyclic relational model  $\mathcal{M}$ , perspective  $B \in \mathcal{E} \cup \mathcal{R}$ , and hop threshold  $h \in \mathbb{N}^0$ , the abstract ground graph  $AGG_{\mathcal{M}Bh}$  is directed and acyclic.*

**Proof.** Let  $\mathcal{M}$  be an arbitrary acyclic relational model, let  $B \in \mathcal{E} \cup \mathcal{R}$  be an arbitrary perspective, and let  $h \in \mathbb{N}^0$  be an arbitrary hop threshold. It is clear that every edge in the abstract ground graph  $AGG_{\mathcal{M}Bh}$  is directed by construction in Definition 5.2. All edges inserted in any abstract ground graph are drawn from the directed dependencies in a relational model. Since  $\mathcal{M}$  is acyclic, they class dependency graph  $G_{\mathcal{M}}$  is also acyclic by Definition 4.9. Assume for contradiction that there exists a cycle of length  $n$  in  $AGG_{\mathcal{M}Bh}$  that contains both relational variables and intersection variables. By Definition 5.2, all edges inserted in  $AGG_{\mathcal{M}Bh}$  are drawn from some dependency in  $\mathcal{M}$ , even for nodes corresponding to intersection variables. Retaining only the final item class in each relational path for every node in the cycle will yield a cycle in  $G_{\mathcal{M}}$  by Definition 4.9. So,  $\mathcal{M}$  could not have been acyclic, which contradicts the assumption. ■

Theorem 5.3 ensures that the standard rules of  $d$ -separation can apply directly to abstract ground graphs because they are acyclic given an acyclic model. We now have sufficient supporting theory to prove that  $d$ -separation on abstract ground graphs is sound and complete. In the following theorem, we define  $\bar{\mathbf{W}}$  as the set of nodes augmented with their corresponding intersection nodes for the set of relational variables  $\mathbf{W}$ :  $\bar{\mathbf{W}} = \mathbf{W} \cup \bigcup_{W \in \mathbf{W}} \{W \cap W' \mid W \cap W' \text{ is an intersection node in } AGG_{\mathcal{M}Bh}\}$ . We also say that  $d$ -separation holds up to a specified hop threshold  $h$  if there are no  $d$ -connecting paths involving a relational path (for some relational variable) of length greater than  $h + 1$ .

**Theorem 5.4** *Relational  $d$ -separation is sound and complete for abstract ground graphs up to a specified hop threshold. Let  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $\mathbf{Z}$  be three distinct sets of relational variables for perspective  $B \in \mathcal{E} \cup \mathcal{R}$  defined over relational schema  $\mathcal{S}$ . Then, for any skeleton  $\sigma$  and for all  $b \in \sigma(B)$ ,  $\mathbf{X}|_b$  and  $\mathbf{Y}|_b$  are  $d$ -separated by  $\mathbf{Z}|_b$  up to hop threshold  $h$  in ground graph  $GG_{\mathcal{M}\sigma}$  if and only if  $\bar{\mathbf{X}}$  and  $\bar{\mathbf{Y}}$  are  $d$ -separated by  $\bar{\mathbf{Z}}$  on the abstract ground graph  $AGG_{\mathcal{M}Bh}$ .*

**Proof.** We must show that  $d$ -separation on an abstract ground graph implies  $d$ -separation on all ground graphs it represents (soundness) and that  $d$ -separation facts that hold across all ground graphs are also entailed by  $d$ -separation on the abstract ground graph (completeness). Let  $\mathcal{M}$  be an arbitrary acyclic relational model, and let  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $\mathbf{Z}$  be three arbitrary distinct sets of relational variables for perspective  $B \in \mathcal{E} \cup \mathcal{R}$  defined over relational schema  $\mathcal{S}$ .

Soundness: Assume that  $\bar{\mathbf{X}}$  and  $\bar{\mathbf{Y}}$  are  $d$ -separated by  $\bar{\mathbf{Z}}$  on  $AGG_{\mathcal{M}Bh}$  for some hop threshold  $h \in \mathbb{N}^0$ . Assume for contradiction that there exists an item  $b \in \sigma(B)$  such that  $\mathbf{X}|_b$  and  $\mathbf{Y}|_b$  are *not*  $d$ -separated by  $\mathbf{Z}|_b$  in the ground graph  $GG_{\mathcal{M}\sigma}$  for some arbitrary skeleton  $\sigma$ . Then, there must exist a  $d$ -connecting path  $p$  from some  $x \in \mathbf{X}|_b$  to some  $y \in \mathbf{Y}|_b$  given all  $z \in \mathbf{Z}|_b$ . By Theorem 5.2,  $AGG_{\mathcal{M}Bh}$  is  $(B, h)$ -reachably complete, so all  $(B, h)$ -reachable edges in  $GG_{\mathcal{M}\sigma}$  are captured by edges in  $AGG_{\mathcal{M}Bh}$ . So, path  $p$  must be represented from all nodes in  $\{n \mid x \in n|_b\}$  to all nodes in  $\{n \mid y \in n|_b\}$  in  $AGG_{\mathcal{M}Bh}$ . If  $p$  is  $d$ -connecting in  $GG_{\mathcal{M}\sigma}$ , then it is  $d$ -connecting in  $AGG_{\mathcal{M}Bh}$ , implying that  $\bar{\mathbf{X}}$  and  $\bar{\mathbf{Y}}$  are *not*  $d$ -separated by  $\bar{\mathbf{Z}}$ . So,  $\mathbf{X}|_b$  and  $\mathbf{Y}|_b$  must be  $d$ -separated by  $\mathbf{Z}|_b$ .

Completeness: Assume that  $\mathbf{X}|_b$  and  $\mathbf{Y}|_b$  are  $d$ -separated by  $\mathbf{Z}|_b$  in the ground graph  $GG_{\mathcal{M}\sigma}$  for all skeletons  $\sigma$  for all  $b \in \sigma(B)$ . Assume for contradiction that  $\bar{\mathbf{X}}$  and  $\bar{\mathbf{Y}}$  are *not*  $d$ -separated by  $\bar{\mathbf{Z}}$  on  $AGG_{\mathcal{M}Bh}$  for some hop threshold  $h \in \mathbb{N}^0$ . Then, there must exist a  $d$ -connecting path  $p$  for some relational variable  $X \in \bar{\mathbf{X}}$  to some  $Y \in \bar{\mathbf{Y}}$  given all  $Z \in \bar{\mathbf{Z}}$ . By Theorem 5.2,  $AGG_{\mathcal{M}Bh}$  is  $(B, h)$ -reachably sound, so every edge in  $AGG_{\mathcal{M}Bh}$  must correspond to some pair of variable instances in some ground graph. So, if  $p$  is  $d$ -connecting in  $AGG_{\mathcal{M}Bh}$ , then there must exist some skeleton  $\sigma$  such that  $p$  is  $d$ -connecting in  $GG_{\mathcal{M}\sigma}$  for some  $b \in \sigma(B)$ , implying that  $d$ -separation does not hold for that ground graph. So,  $\bar{\mathbf{X}}$  and  $\bar{\mathbf{Y}}$  must be  $d$ -separated by  $\bar{\mathbf{Z}}$  on  $AGG_{\mathcal{M}Bh}$ . ■

Theorem 5.4 proves that  $d$ -separation on abstract ground graphs is a sound and complete solution to identifying independence in relational models. Given Theorem 5.1, we can also say that the set of conditional independence facts derived from  $d$ -separation on abstract ground graphs is exactly the same (up to a specified hop threshold) as the set of conditional independencies in common with all faithful distributions represented by all possible ground graphs.

Additionally, we can show that relational  $d$ -separation is equivalent to the Markov condition on relational models. In the following definition, let  $nd(X)$  be the non-descendant variables of  $X$  and let  $pa(X)$  be the set of parent variables of  $X$ .

**Definition 5.5 (Relational Markov condition)** Let  $X = [B, \dots, I_k].V$  be a relational variable for perspective  $B \in \mathcal{E} \cup \mathcal{R}$  defined over relational schema  $\mathcal{S}$ . Then, for relational model  $\mathcal{M}$ ,  $P(X \mid nd(X), pa(X)) = P(X \mid pa(X))$  if and only if, for any skeleton  $\sigma$ ,  $\forall x \in X|_b$   $P(x \mid nd(x), pa(x)) = P(x \mid pa(x))$  in ground graph  $GG_{\mathcal{M}\sigma}$  for all  $b \in \sigma(B)$ .



In other words, a relational variable  $X$  is independent of its non-descendants given its parents if and only if, for any possible ground graph, the Markov condition holds for all instances of  $X$ . For Bayesian networks, the Markov condition is equivalent to  $d$ -separation (Neapolitan, 2004). Because ground graphs are Bayesian networks (by Lemma 5.3) and relational  $d$ -separation on abstract ground graphs is sound and complete (by Theorem 5.4), we can conclude that relational  $d$ -separation is equivalent to the relational Markov condition.

## 6. Naïve Relational $d$ -separation is Frequently Incorrect

If the rules of  $d$ -separation for Bayesian networks were applied directly to relational models, how frequently would the conditional independence conclusions be correct? In this section, we evaluate the necessity of our approach to relational  $d$ -separation, namely the use of abstract ground graphs. We empirically test the consistency of a naïve approach to relational  $d$ -separation against our sound and complete solution over a large space of synthetic causal models. In order to do so, we first formally define how  $d$ -separation could actually be applied directly to the structure of a relational model. Consider the following limited definition of relational paths, which itself limits the space of models and conditional independence queries.

**Definition 6.1 (Simple relational path)** A *simple relational path*  $[I_1, \dots, I_k]$  for relational schema  $\mathcal{S}$  is an alternating sequence of entity and relationship classes  $I_1, \dots, I_k \in \mathcal{E} \cup \mathcal{R}$  such that  $I_1 \neq \dots \neq I_k$  and for all  $j > 1$  (1) if  $I_j \in \mathcal{E}$ , then  $I_{j-1} \in \mathcal{R}$  with  $I_j \in I_{j-1}$  and (2) if  $I_j \in \mathcal{R}$ , then  $I_{j-1} \in \mathcal{E}$  with  $I_{j-1} \in I_j$ .

The sole difference between relational paths (Definition 4.3) and simple relational paths is that no item class may appear multiple times along a simple relational path. This yields paths drawn directly from a schema diagram. For the example in Figure 1(a), [EMPLOYEE, DEVELOPS, PRODUCT] is simple whereas [EMPLOYEE, DEVELOPS, PRODUCT, DEVELOPS, EMPLOYEE] is not.

Additionally, we define *simple relational schemas* such that for any two item classes  $I_j, I_k \in \mathcal{E} \cup \mathcal{R}$ , there exists at most one simple relational path between them, that is, no cycles occur in the schema diagram. The example in Figure 1(a) is a simple relational schema. The restriction to simple relational paths and schemas yields similar definitions for *simple relational variables*, *simple relational dependencies*, and *simple relational models*. The example in Figure 2(a) is a simple relational model because it consists of only simple relational dependencies.

A first approximation to relational  $d$ -separation would be to apply the rules of  $d$ -separation for Bayesian networks directly to the graphical representation of relational models. This is equivalent to constructing the class dependency graph of Definition 4.9 from the relational model and applying  $d$ -separation. The class dependency graph for the model in Figure 2(a) is shown in Figure 8(a). Note that the class dependency graph removes perspectives, ignores path designators on dependencies, does not include all implications of dependencies among arbitrary relational variables, and does not consider intersection variables.

Although the class dependency graph is independent of perspectives, testing any conditional independence fact *requires* choosing a perspective. All variables must have a common

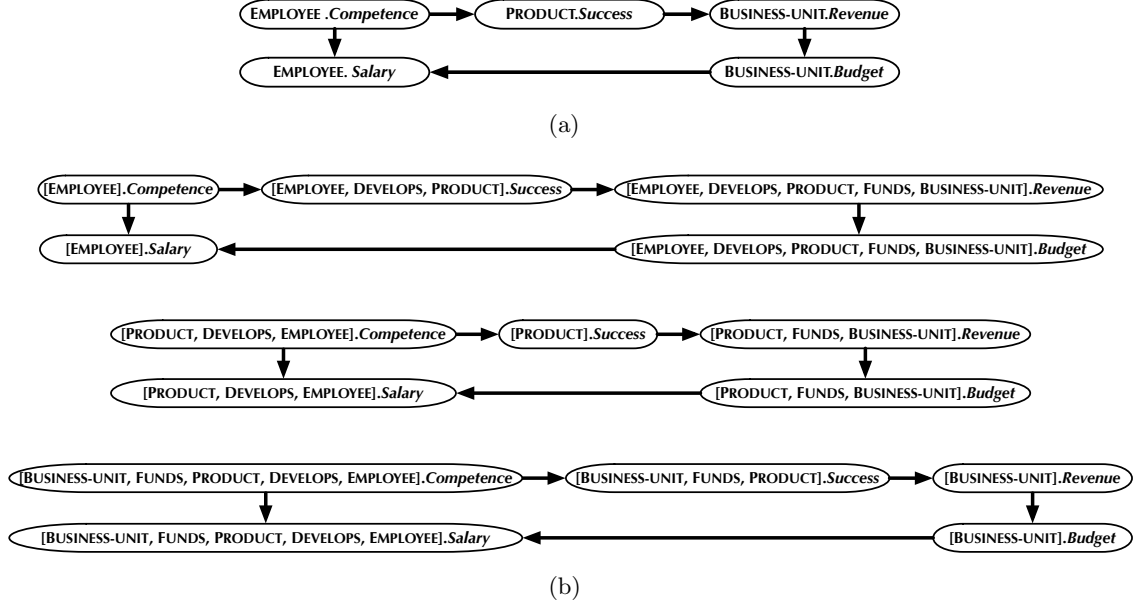


Figure 8: For the model in Figure 2(a), (a) the class dependency graph and (b) three simple abstract ground graphs for the EMPLOYEE, PRODUCT, and BUSINESS-UNIT perspectives.

base item; otherwise, no method can produce a single consistent, propositional table from a relational database. For example, consider the construction of a table describing employees with columns for their salary, the success of products they develop, and the revenue of the business units they operate under. This procedure requires a join across three relational variables (`[EMPLOYEE].Salary`, `[EMPLOYEE, DEVELOPS, PRODUCT].Success`, and `[EMPLOYEE, DEVELOPS, PRODUCT, FUNDS, BUSINESS-UNIT].Revenue`) for every common base item instance, from Paul to Thomas. Therefore, given a perspective and the space of simple relational schemas and models, a class dependency graph is equivalent to a *simple abstract ground graph*.

**Definition 6.2 (Simple abstract ground graph)** For simple relational model  $\mathcal{M} = (\mathcal{S}, \mathcal{D})$ , perspective  $B \in \mathcal{E} \cup \mathcal{R}$ , and hop threshold  $h \in \mathbb{N}^0$ , the *simple abstract ground graph*  $AGG_{\mathcal{M}Bh}^s$  is the directed acyclic graph  $(V, E)$  that abstracts the dependencies  $\mathcal{D}$  among simple relational variables. The set of nodes is the union of all simple relational variables  $\{[B, \dots, I_j].V \mid B \neq \dots \neq I_j \wedge \text{length}([B, \dots, I_j]) \leq h + 1\}$ , and the set of edges connects simple relational variables  $\{[B, \dots, I_k].V_1 \rightarrow [B, \dots, I_j].V_2 \mid [I_j, \dots, I_k].V_1 \rightarrow [I_j].V_2 \in \mathcal{D} \wedge [B, \dots, I_k] \in \text{extend}([B, \dots, I_j], [I_j, \dots, I_k]) \wedge [B, \dots, I_k].V_1, [B, \dots, I_j].V_2 \in V\}$ .

Simple abstract ground graphs only include nodes for simple relational variables and necessarily exclude intersection variables. Lemma 4.1 only applies to pairs of simple relational paths if the schema has cycles, which is not the case for simple relational schemas. As a result, for all possible perspectives of a given schema and model, all simple abstract

ground graphs consist of the same number of nodes and edges when the hop threshold is sufficiently large and with the path designator on relational variables redefined from the given perspective. Figure 8(b) shows three simple abstract ground graphs from unique perspectives for the model in Figure 2(a). As noted above, simple abstract ground graphs are qualitatively the same as the class dependency graph, but this representation enables answering relational  $d$ -separation queries, which requires a common perspective.

**Remark 6.1** The naïve approach to relational  $d$ -separation derives conditional independence facts from simple abstract ground graphs (Definition 6.2). The sound and complete approach described in this paper applies  $d$ -separation, augmenting the variable sets with their intersection variables, to “regular” abstract ground graphs, as described by Definition 5.2. Clearly, if  $d$ -separation on a simple abstract ground graph claims that  $\mathbf{X}$  is  $d$ -separated from  $\mathbf{Y}$  given  $\mathbf{Z}$ , then  $d$ -separation on the regular abstract ground graph yields the same conclusion if and only if there are no  $d$ -connecting paths on the regular abstract ground graph. Either  $\mathbf{X}$  and  $\mathbf{Y}$  can be  $d$ -separated by a set of simple relational variables  $\mathbf{Z}$ , or they require non-simple relational variables—those involving relational paths with cycles or intersection variables.<sup>8</sup>  $\square$

To evaluate the necessity of regular abstract ground graphs (i.e., the additional paths involving non-simple relational variables and intersection variables), we devised an experiment to determine the frequency of equivalence of  $d$ -separation on simple and regular abstract ground graphs. The two approaches are only equivalent if a minimal separating set consists entirely of simple relational variables. Thus, for a given pair of relational variables  $X$  and  $Y$ , we test the following:

1. Is either  $X$  or  $Y$  a non-simple relational variable?
2. Are  $X$  and  $Y$  marginally independent?
3. Does a minimal separating set  $\mathbf{Z}$   $d$ -separate  $X$  and  $Y$ , where  $\mathbf{Z}$  consists solely of simple relational variables?
4. Is there no separating set  $\mathbf{Z}$  that  $d$ -separates  $X$  and  $Y$ ?

If the answer to (1) is yes, then the naïve approach cannot apply since either  $X$  or  $Y$  is not defined on the simple abstract ground graph. If the answer to (2) is yes, then there is equivalence; this is a trivial case because there are no  $d$ -connecting paths for  $\mathbf{Z} = \emptyset$ . If the answer to (3) is yes, then there is a minimal separating set  $\mathbf{Z}$  consisting of only simple relational variables. In this case, the simple abstract ground graph is sufficient, and we have equivalence. If the answer to (4) is yes, then no separating set  $\mathbf{Z}$ , simple or otherwise, renders  $X$  and  $Y$  conditionally independent.

We randomly generated a schema and model for 100 trials for each setting using the following parameters:

---

8. The theoretical conditions under which equivalence occurs are sufficiently complex that they provide little utility as they essentially require reconstructing the regular abstract ground graph and checking a potentially exponential number of dependency paths.

- Number of entity classes, ranging from 1 to 4.
- Number of relationship classes fixed to one less than the number of entities, allowing only simple relational schemas. Relationship cardinalities are randomly selected.
- Number of attributes for each entity and relationship class randomly drawn from a shifted Poisson distribution with mean  $\lambda = 1.0$  ( $\sim \text{Pois}(1.0) + 1$ ).
- Number of dependencies in the model, ranging from 1 to 10.

Then, for all pairs of relational variables with a common perspective for hop threshold  $h = 4$ , we ran the aforementioned tests against the abstract ground graph with hop threshold  $h = 8$ .

This procedure generated a total of almost 3.6 million pairs of relational variables to test. Approximately 56% included a non-simple relational variable; the naïve approach cannot be used to derive a conditional independence statement in these cases, requiring the full abstract ground graph in order to represent these variables. Of the remaining 44% (roughly 1.6 million), 82% were marginally independent, and 9% were not conditionally independent given any conditioning set  $\mathbf{Z}$ . Then, of the remaining 9% (roughly 145,000), we can test the frequency of equivalence for conditional independence facts with non-empty separating sets—the percentage of cases for which only simple relational variables are required in a minimal separating set  $\mathbf{Z}$ .

Figure 9 shows this frequency for schemas of increasing numbers of entity classes (1–4) for varying numbers of dependencies in the causal model (1–10). Since relational schemas with a single entity class and no relationships describe propositional data, the simple abstract ground graph is equivalent to the full abstract ground graph, which is also equivalent to the model itself. In this case, the naïve approach is always equivalent because it is exactly  $d$ -separation on Bayesian networks. For truly relational schemas (with more than one entity class and at least one relationship class), the frequency of equivalence decreases as the number of dependencies in the model increases. Additionally, the frequency of equivalence decreases more as the number of entities in the schema increases. For example, the frequency of equivalence for nine dependencies is 60.3% for two entities, 51.2% for three entities, and 43.2% for four entities.

There are several factors that influence the frequency of equivalence. We learned a linear regression model with interaction terms and indicator variables for the number of entities, dependencies, and MANY cardinalities on relationships. In predicting the raw number of non-equivalent conditional independencies (those requiring non-simple relational variables in the separating set), the strongest predictor is the interaction between the number of MANY cardinalities and the number of dependencies. This suggests that the more dependencies in the model and the more MANY cardinalities, the more non-equivalent conditional independencies will occur. Thus, the frequency of equivalence illustrated in Figure 9 is only so high because the experiment averages across randomly chosen schemas with randomly selected relationship cardinalities. When there are few MANY cardinalities, the model can be more closely approximated by a Bayesian network, for which we expect the naïve approach to be more equivalent. In contrast, in predicting the raw number of *equivalent* conditional independencies (those with only simple relational variables in the separating

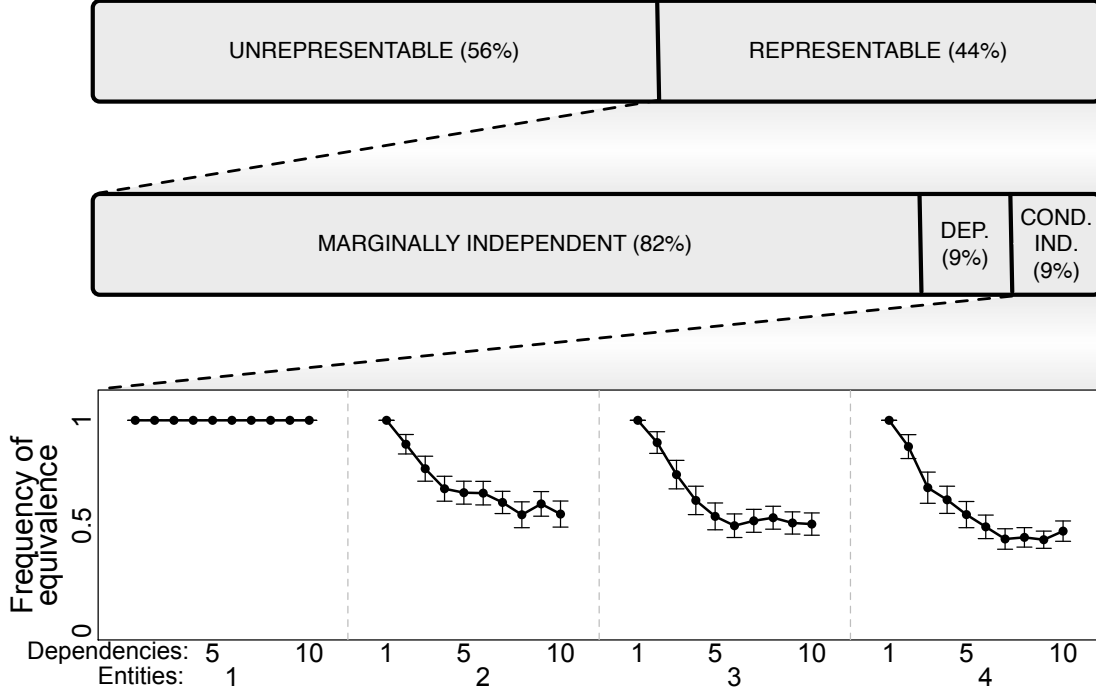


Figure 9: Of all generated relational  $d$ -separation queries, 56% are not representable with the naïve approach. Of the 44% that are representable (involving only simple relational variables), 82% are marginally independent and 9% are dependent. Pairs of relational variables in the remaining 9% are conditionally independent given a non-empty separating set ( $X \perp\!\!\!\perp Y \mid \mathbf{Z}$ , where  $\mathbf{Z} \neq \emptyset$ ). We test whether the *conditioning set* consists solely of simple relational variables. If so, then the naïve approach to relational  $d$ -separation is equivalent to  $d$ -separation on fully specified abstract ground graphs. This graph plots to the frequency of equivalence across schemas with increasing numbers of entity classes (1–4) for varying numbers of dependencies (1–10). For schemas with more than one entity class, the frequency of equivalence decreases as the number of dependencies increases. Shown with 95% confidence intervals.

set), the strongest predictors are non-interaction terms: the number of entities (positive), the number of dependencies (positive), and the number of MANY cardinalities (negative). Conditional independencies occur more frequently when there are more entities and dependencies; however, non-simple relational variables occur more frequently when there are more MANY cardinalities.

This experiment suggests that applying  $d$ -separation directly to a relational model will frequently derive incorrect conditional independence facts. Additionally, there is a large class of conditional independence queries involving non-simple parent or child variables for which such an approach is undefined. Together, this indicates that fully specifying

abstract ground graphs and applying  $d$ -separation augmented with intersection variables (as described in Section 5) is critical for accurately deriving most conditional independence facts from relational models.

## 7. Experiments

To complement the theoretical results, we present a series of three experiments on synthetic data. The primary goal of these empirical results is to demonstrate the feasibility of applying relational  $d$ -separation in practice. The experiment in Section 7.1 describes the factors that influence the size of abstract ground graphs and thus the computational complexity of relational  $d$ -separation. The experiment in Section 7.2 evaluates the growth rate of separating sets produced by relational  $d$ -separation as abstract ground graphs become large. The results indicate that minimal separating sets grow more slowly than abstract ground graphs. The experiment in Section 7.3 tests how the expectations of the relational  $d$ -separation theory match statistical conclusions on simulated data. As expected from the proofs of correctness in Section 5.2, the results indicate a close match, aside from Type I errors and certain biases of conventional statistical tests on relational data.

### 7.1 Abstract Ground Graph Size

Abstract ground graphs (AGGs) explicitly represent the intersection among relational variables and extend the canonically specified dependencies of relational models. Consequently, it is important to quantify the size of an AGG and determine which factors influence its size. We randomly generated relational schemas and models for 1,000 trials of each setting using the following parameters:

- Number of entity classes, ranging from 1 to 4.
- Number of relationship classes, ranging from 0 to 4. The schema is guaranteed to be fully connected (e.g., schemas with three entity classes require at least two relationships) and includes at most a single relationship between a pair of entities (e.g., schemas with two entity classes must have exactly one relationship class). Relationship cardinalities are selected uniformly at random.
- Number of attributes for each entity and relationship class randomly drawn from a shifted Poisson distribution with mean  $\lambda = 1.0$  ( $\sim \text{Pois}(1.0) + 1$ ).
- Number of dependencies in the model, ranging from 1 to 15.

This procedure generated a total of 450,000 abstract ground graphs across all perspectives, including all entity and relationship classes for each experimental combination. We measure size as the number of nodes and edges appearing in a given abstract ground graph.

Figure 10(a) depicts how AGG size varies with respect to the number of MANY cardinalities in the schema (fixed for models with 10 dependencies), and Figure 10(b) shows how it varies with respect to the number of dependencies in the model. Note that for a single entity, abstract ground graphs are equivalent to Bayesian networks. In order to determine the most influential factors of AGG size, we ran log-linear regression using independent

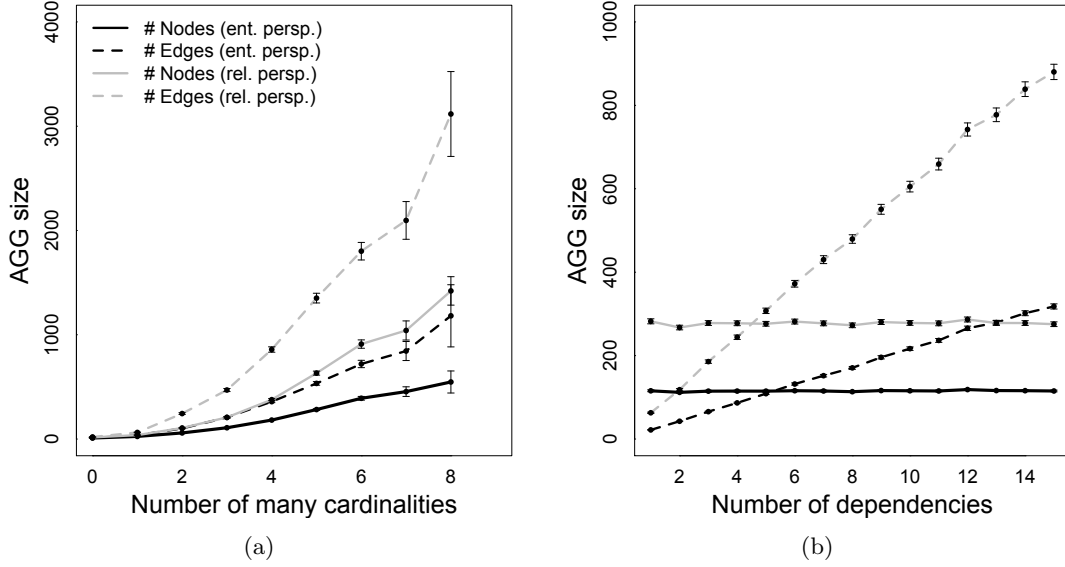


Figure 10: AGG size variation as (a) the number of MANY cardinalities in the schema increases (dependencies fixed at 10) and (b) the number of dependencies increases. Shown with 95% confidence intervals.

variables that describe only the schema and model. Detailed results are provided in Appendix C. This analysis indicates: (1) As the number of entities, relationships, attributes, and MANY cardinalities increases, the AGG grows at an exponential rate with respect to both nodes and edges. (2) As the number of dependencies in the model increases, the number of edges increases linearly, but the number of nodes is invariant. (3) AGGs with relationship perspectives are larger than entity perspectives because more relational variables can be defined.

## 7.2 Minimal Separating Set Size

Because abstract ground graphs can become large, one might expect that separating sets<sup>9</sup> would also grow to impractical sizes. Fortunately, relational  $d$ -separation produces minimal separating sets that are empirically observed to be small. We ran 1,000 trials of each setting using the following parameters:

- Number of entity classes, ranging from 1 to 4.
- Number of relationship classes fixed to one less than the number of entities. Relationship cardinalities are selected uniformly at random.
- Total number of attributes across entity and relationship classes fixed to 10.
- Number of dependencies in the model, ranging from 1 to 10.

9. If  $\mathbf{X}$  and  $\mathbf{Y}$  are  $d$ -separated given  $\mathbf{Z}$ , then  $\mathbf{Z}$  is a separating set for  $\mathbf{X}$  and  $\mathbf{Y}$ . A separating set  $\mathbf{Z}$  is *minimal* if there is no proper subset of  $\mathbf{Z}$  that is also a separating set.

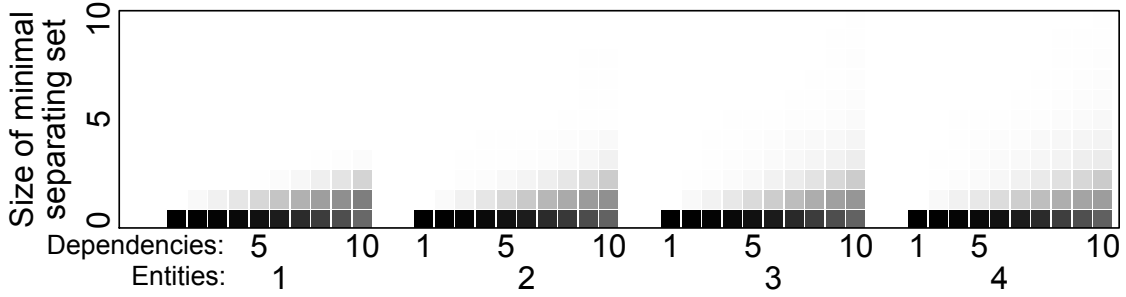


Figure 11: Minimal separating sets have reasonable sizes, growing only with respect to model density.

For each relational model, we identified one minimal separating set for up to 100 randomly chosen pairs of conditionally independent relational variables. This procedure generated almost 2.5 million pairs of variables.

To identify a minimal separating set between relational variables  $X$  and  $Y$ , we modified Algorithm 4 devised by Tian et al. (1998) by starting with all parents of  $\bar{X}$  and  $\bar{Y}$ , the variables augmented with the intersection variables they subsume in the abstract ground graph. While the discovered separating sets are *minimal*, they are not necessarily of *minimum* size. Figure 11 shows the frequency of separating set size as both the number of entities and dependencies vary. In summation, roughly 83% are marginal independencies (having empty separating sets), 13% have separating sets of size 1, and less than 0.1% have separating sets with more than 5 variables. The experimental results indicate that separating set size is strongly influenced by model density, primarily because the number of potential  $d$ -connecting paths increases as the number of dependencies increases.

### 7.3 Empirical Validity

As a practical demonstration, we examined how the expectations of the relational  $d$ -separation theory match the results of statistical tests on actual data. We parameterized relational models in a way that produced linear effects with equal contribution from each direct cause and standard methods for aggregating terminal sets of relational variables. We used simple additive linear equations with independent, normally distributed error, and the average aggregate for terminal sets of relational variables. If  $Y$  has no parents, then  $Y = \epsilon \sim N(0, 1)$ , and  $Y = \sum_{X \in \text{par}(Y)} \frac{0.9}{|\text{par}(Y)|} \text{avg}(X) + .1\epsilon$  otherwise. Relational skeletons

were constructed such that relationship instances were produced via a latent homophily process, similar to the method used by Shalizi and Thomas (2011). Each entity instance received a single latent variable, marginally independent from all other variables. The probability of any relationship instance was drawn from  $\frac{e^{-\alpha d}}{1 + e^{-\alpha d}}$ , the inverse logistic function, where  $d = |L_{E_1} - L_{E_2}|$ , the difference between the latent variables on the two entities, and  $\alpha = 10$ , set as the decay parameter. We also scaled the probabilities in order to produce



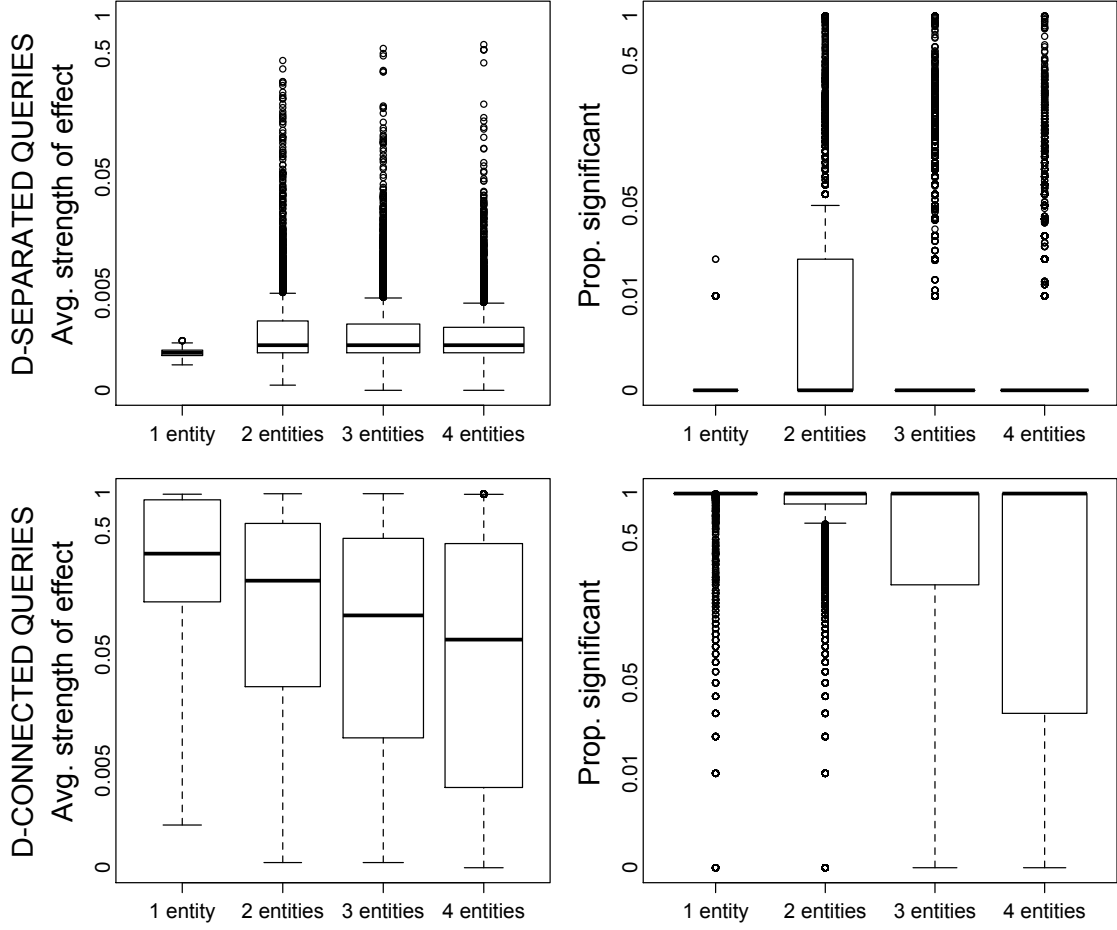


Figure 12: The relational  $d$ -separation theory closely matches the results of statistical tests on actual data.

an expected degree of 5 for each entity instance when the cardinality of the relationship is MANY. Since the latent variables are marginally independent of all others, they are safely omitted from abstract ground graphs; their sole purpose was to generate relational skeletons, providing a greater probability of non-empty intersection variables as opposed to a random underlying link structure.

For 100 trials, we randomly generated a relational schema and model using the following parameter settings:

- Number of entity classes, ranging from 1 to 4.
- Number of relationship classes fixed to one less than the number of entities. Relationship cardinalities are selected uniformly at random.
- Number of attributes for each entity and relationship class randomly drawn from a shifted Poisson distribution with mean  $\lambda = 1.0$  ( $\sim Pois(1.0) + 1$ ).

- Number of dependencies in the model fixed to 10.

We then tested up to 100 true and false relational  $d$ -separation queries across 100 skeletons (i.e., instantiated relational databases) with 1,000 instances of each entity class. To test for conditional independence between  $X$  and  $Y$  given  $\mathbf{Z}$  ( $X \perp\!\!\!\perp Y \mid \mathbf{Z} ?$ ), we tested the  $t$ -statistic for the coefficient of  $avg(X)$  in the equation  $Y = \beta_0 + \beta_1 avg(X) + \sum_{Z_i \in \mathbf{Z}} \beta_i avg(Z_i)$ .

The queries were restricted such that  $X$  and  $Y$  are single relational variables, and the relational path for  $Y$  is a singleton. These queries correspond to testing potential canonical (direct causal) dependencies.

For each query, we recorded the average strength of effect (as measured by the squared partial correlation coefficient—the proportion of remaining variance of  $Y$  explained by  $X$  after conditioning on  $\mathbf{Z}$ ) and the proportion of trials for which each query was significant ( $\alpha = 0.01$  adjusted with Bonferroni correction with the number of queries per trial). Figure 12 shows the distribution of the average strength of effect and proportion of significant trials across both true and false queries for varying numbers of entities. The graph uses a standard box-and-whisker plot with values greater or less than 1.5 times the inner quartile range—the difference between the upper and lower quartiles—marked as outliers.

In the vast majority of cases, relational  $d$ -separation is consistent with tests on actual data. For approximately 23,000 true queries, 14.9% are significant in more than one trial, but only 2.2% have an average effect size greater than 0.01. Aside from Type I error, a small number of cases exhibit an interaction between aggregation and relational structure (i.e., degree or the cardinality of terminal sets of relational variables), and even small effects produce significant results with a large sample size. This interaction violates the identically distributed assumption of data instances, which produces a biased estimate of effect size for simple linear regression. Linear regression does not account for these interaction effects, suggesting the need for more accurate statistical tests of conditional independence for relational data.

## 8. Discussion

In this paper, we extend the theory of  $d$ -separation to graphical models of relational data. We present the *abstract ground graph*, a new representation that is  $(B, h)$ -reachably sound and complete in its abstraction of dependencies across all possible ground graphs of a given relational model. We formally define relational  $d$ -separation and offer a sound, complete, and computationally efficient approach to deriving conditional independence facts from relational models by exploiting their abstract ground graphs. We also show that relational  $d$ -separation is equivalent to the Markov condition for relational models. We provide an empirical analysis of relational  $d$ -separation on synthetic data, demonstrating a close correspondence between the theory and statistical results in practice. Finally, we evaluate how frequently the complexity of abstract ground graphs proves necessary for accurately deriving conditional independence facts.

The results of this paper imply potential flaws in the design and analysis of some real-world studies. If researchers of social or economic systems choose inappropriate data and model representations, then their analyses may omit important classes of dependencies. Specifically, our theory implies that choosing a propositional representation from an in-

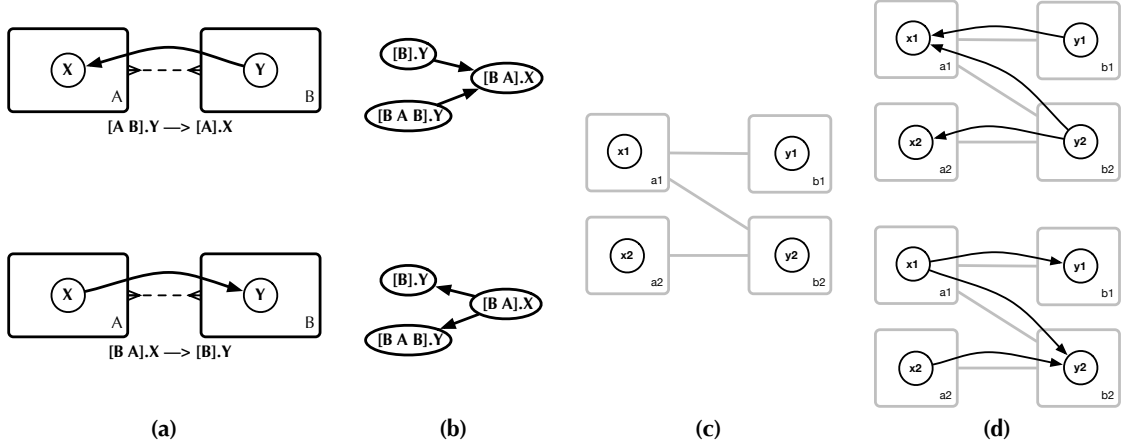


Figure 13: (a) Two models of a bivariate relational domain with opposite directions of causality for a single dependency (relationship class omitted for simplicity); (b) a single dependency implies additional dependencies among arbitrary relational variables, shown here in a fragment of the abstract ground graph for  $B$ 's perspective; (c) an example relational skeleton; and (d) the ground graphs resulting from applying the relational model to the skeleton.

herently relational domain may lead to serious errors. An abstract ground graph from a given perspective defines the exact set of variables that must be included in any propositionalization. The absence of any relational variable (including intersection variables) may unnecessarily violate causal sufficiency, which could result in the inference of a causal dependency where conditional independence was not detected. Our work indicates that researchers should carefully consider how to represent their domains in order to accurately reason about conditional independence.

The abstract ground graph representation also presents an opportunity to derive new edge orientation rules for algorithms that learn the structure of relational models, such as RPC (Maier et al., 2010). There are unique orientations of edges that are consistent with a given pattern of association that can only be recognized in an abstract ground graph. For example, in contrast to bivariate IID data, it is simple to establish the direction of causality for bivariate relational data. Consider the two bivariate, two-entity relational models depicted in Figure 13(a). The first model implies that values of  $X$  on  $A$  entities are caused by the values of  $Y$  on related  $B$  entities. The second model implies the opposite, that values of  $Y$  on  $B$  entities are caused by the values of  $X$  on related  $A$  entities. For simplicity, we show the relationship class only as a dashed line between entity classes and omit it from relational paths.

Figure 13(b) illustrates a fragment of the abstract ground graph (for hop threshold  $h=4$ ) that each of the two relational models implies. As expected, the directions of the edges in the two abstract ground graphs are counterposed. Both models produce observable statistical dependencies for relational variable pairs  $\langle [B].Y, [B\ A].X \rangle$  and  $\langle [B\ A].X, [B\ A\ B].Y \rangle$ . However, the relational variables  $[B].Y$  and  $[B\ A\ B].Y$  have different observable statisti-

cal dependencies: In the first model, they are marginally independent and conditionally dependent given  $[B, A].X$ , and in the second model, they are marginally dependent and conditionally independent given  $[B, A].X$ . As a result, we can uniquely determine the direction of causality of the single dependence by exploiting relational structure. (There is symmetric reasoning for relational variables from  $A$ 's perspective, and this result is also applicable to ONE-to-MANY data.)

To illustrate this fact more concretely, consider the small relational skeleton shown in Figure 13(c) and the ground graphs applied to this skeleton in Figure 13(d). In the first ground graph, we have  $y_1 \perp\!\!\!\perp y_2$  and  $y_1 \not\perp\!\!\!\perp y_2 | x_1$ , but in the second ground graph, we have  $y_1 \not\perp\!\!\!\perp y_2$  and  $y_1 \perp\!\!\!\perp y_2 | x_1$ . These opposing conditional independence relations uniquely determine the correct causal model.

Deriving and formalizing the implications of relational  $d$ -separation is a main direction of future research. Additionally, our experiments suggest that more accurate tests of conditional independence for relational data need to be developed, specifically tests that can address the interaction between relational structure and aggregation across terminal sets of relational variables. This work has also focused solely on relational models of attributes; future work should consider models of relationship and entity existence to fully characterize generative models of relational structure. Finally, the theory could also be extended to incorporate functional or deterministic dependencies, as  $D$ -separation extends  $d$ -separation for Bayesian networks.

## Acknowledgments

The authors wish to thank Cindy Loiselle for her editing expertise. This effort is supported by the Intelligence Advanced Research Project Agency (IARPA) via Department of Interior National Business Center Contract number D11PC20152, Air Force Research Lab under agreement number FA8750-09-2-0187 and Science Applications International Corporation (SAIC) and DARPA under contract number P010089628. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of IARPA, DoI/NBC, AFRL, SAIC, DARPA or the U.S. Government. Katerina Marazopoulou received scholarship support from the Greek State Scholarships Foundation.

## Appendix A.

In this appendix, we provide detailed proofs for all previous lemmas, theorems, and corollaries.

**Lemma 4.1** *For any schema  $\mathcal{S}$  and any two relational paths  $P_1 = [I_1, \dots, I_m, \dots, I_k]$  and  $P_2 = [I_1, \dots, I_n, \dots, I_k]$  with  $I_m \neq I_n$ , there exists a skeleton  $\sigma$  such that  $P_1|_{i_1} \cap P_2|_{i_1} \neq \emptyset$  for some  $i_1 \in \sigma(I_1)$ .*

**Proof.** Proof by construction. Let  $\mathcal{S}$  be an arbitrary schema with two arbitrary relational paths  $P_1 = [I_1, \dots, I_m, \dots, I_k]$  and  $P_2 = [I_1, \dots, I_n, \dots, I_k]$  where  $I_m \neq I_n$ . We will con-

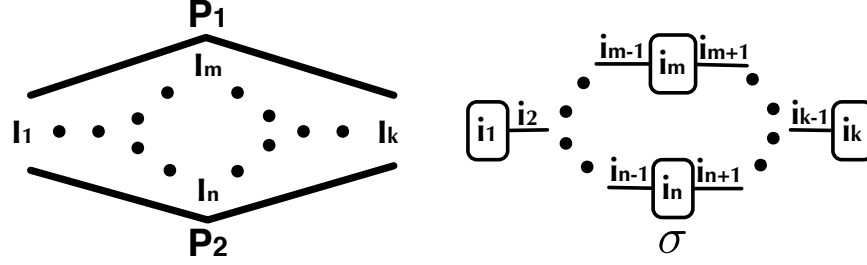


Figure 14: Schematic of two relational paths  $P_1$  and  $P_2$  for which Lemma 4.1 guarantees that some skeleton  $\sigma$  yields a nonempty intersection of their terminal sets. The example depicts a possible constructed skeleton based on the procedure used in the proof of Lemma 4.1.

construct a skeleton  $\sigma$  such that the terminal sets for item  $i_1 \in \sigma(I_1)$  along  $P_1$  and  $P_2$  have a nonempty intersection, that is, an item  $i_k \in P_1|_{i_1} \cap P_2|_{i_1} \neq \emptyset$  (roughly depicted in Figure 14). We use the following procedure to build  $\sigma$ :

1. Simultaneously traverse  $P_1$  and  $P_2$  from  $I_1$  until the paths diverge. For each entity class  $E \in \mathcal{E}$  reached, add a unique entity instance  $e$  to  $\sigma(E)$ .
2. Simultaneously traverse  $P_1$  and  $P_2$  backwards from  $I_k$  until the paths diverge. For each entity class  $E \in \mathcal{E}$  reached, add a unique entity instance  $e$  to  $\sigma(E)$ .
3. For the divergent subpaths of both  $P_1$  and  $P_2$ , add unique entity instances for each entity class  $E \in \mathcal{E}$ .
4. Repeat 1–3 for relationship classes. For each  $R \in \mathcal{R}$  reached, add a unique relationship instance  $r$  connecting the entity instances from classes on  $P_1$  and  $P_2$ , and add unique entity instances for classes  $E \in R$  not appearing on  $P_1$  and  $P_2$ .

This process constructs an admissible skeleton—all instances are unique and this process assumes no cardinality constraints aside from those required by Definition 4.3. By construction, there exists an item  $i_1 \in \sigma(I_1)$  such that  $P_1|_{i_1} \cap P_2|_{i_1} = \{i_k\} \neq \emptyset$ . ■

**Lemma 5.1** *For any two relational paths  $P_{orig} = [I_1, \dots, I_j]$  and  $P_{ext} = [I_j, \dots, I_k]$  with  $\mathbf{P} = \text{extend}(P_{orig}, P_{ext})$ ,  $\forall P \in \mathbf{P}$  there exists a relational skeleton  $\sigma$  such that  $\exists i_1 \in \sigma(I_1)$  such that  $\exists i_k \in P|_{i_1}$  and  $\exists i_j \in P_{orig}|_{i_1}$  such that  $i_k \in P_{ext}|_{i_j}$ .*

**Proof.** Let  $P_{orig} = [I_1, \dots, I_j]$  and  $P_{ext} = [I_j, \dots, I_k]$  be two arbitrary relational paths, and let  $\mathbf{P} = \text{extend}(P_{orig}, P_{ext})$ . Let  $c \in \text{pivots}(\text{reverse}(P_{orig}), P_{ext})$  be an arbitrary pivot such that  $P = \text{concat}(P_{orig}[0 : \text{length}(P_{orig}) - c + 1], P_{ext}[c : \text{length}(P_{ext})])$ . Since  $P \in \mathbf{P}$ , we know that  $P$  is a valid relational path, and we can ignore pivots that produce invalid paths. There are two subcases:

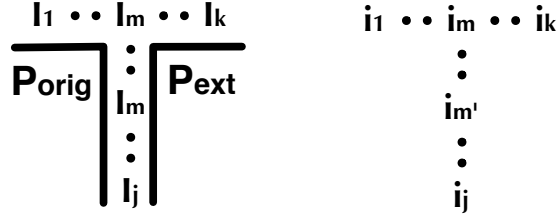


Figure 15: Example construction of a relational skeleton for two relational paths  $P_{orig} = [I_1, \dots, I_m, \dots, I_j]$  and  $P_{ext} = [I_j, \dots, I_m, \dots, I_k]$ , where item class  $I_m$  is repeated between  $I_m$  and  $I_j$ . This construction is used within the proof of Lemma 5.1.

(a)  $c = 1$  and  $P = [I_1, \dots, I_j, \dots, I_k]$ . This subcase holds generally for any skeleton. Proof by contradiction. Let  $\sigma$  be an arbitrary skeleton, choose  $i_1 \in \sigma(I_1)$  arbitrarily, and choose  $i_k \in P|_{i_1}$  arbitrarily. Assume for contradiction that there is no  $i_j$  in the terminal set  $P_{orig}|_{i_1}$  such that  $i_k$  would be in the terminal set  $P_{ext}|_{i_j}$ , that is,  $\forall i_j \in P_{orig}|_{i_1} \ i_k \notin P_{ext}|_{i_j}$ . Since  $P = [I_1, \dots, I_j, \dots, I_k]$ , we know that  $i_k$  is reached by traversing  $\sigma$  from  $i_1$  via some  $i_j$  to  $i_k$ . But the traversal from  $i_1$  to  $i_j$  implies that  $i_j \in [I_1, \dots, I_j]|_{i_1} = P_{orig}|_{i_1}$ , and the traversal from  $i_j$  to  $i_k$  implies that  $i_k \in [I_j, \dots, I_k]|_{i_j} = P_{ext}|_{i_j}$ . So, there must exist an  $i_j \in P_{orig}|_{i_1}$  such that  $i_k \in P_{ext}|_{i_j}$ .

(b)  $c > 1$  and  $P = [I_1, \dots, I_m, \dots, I_k]$ . Proof by construction. We build a relational skeleton  $\sigma$  following the same procedure as outlined in the proof of Lemma 4.1. Add instances to  $\sigma$  for every item class that appears on  $P_{orig}$  and  $P_{ext}$ . Since  $P = [I_1, \dots, I_m, \dots, I_k]$ , we know that  $i_k$  is reached by traversing  $\sigma$  from  $i_1$  via some  $i_m$  to  $i_k$ . By case (a),  $\exists i_m \in [I_1, \dots, I_m]|_{i_1}$  such that  $i_k \in [I_m, \dots, I_k]|_{i_m}$ . We then must show that there exists an  $i_j \in [I_m, \dots, I_j]|_{i_m}$  with  $i_m \in [I_j, \dots, I_m]|_{i_j}$ . But, constructing the skeleton with unique item instances for every appearance of an item class on the relational paths provides this and does not violate cardinality constraints. If any item class appears more than once, then the bridge burning semantics are induced. However, adding an additional item instance for every reappearance of item classes enables the traversal from  $i_j$  to  $i_m$  and vice versa. An example of this construction is displayed in Figure 15. This is also a valid relational skeleton because  $P_{orig}$  and  $P_{ext}$  are valid relational paths, and by definition the cardinality constraints of the schema must permit multiple (MANY) instances in the skeleton of any repeated item class. By this procedure, we show that there exists a skeleton  $\sigma$  such that there exists an  $i_j \in P_{orig}|_{i_1}$  such that  $i_k \in P_{ext}|_{i_j}$ . ■

**Lemma 5.2** *For any relational skeleton  $\sigma$  and two relational paths  $P_{orig} = [I_1, \dots, I_j]$  and  $P_{ext} = [I_j, \dots, I_k]$  with  $\mathbf{P} = \text{extend}(P_{orig}, P_{ext})$ ,  $\forall i_1 \in \sigma(I_1) \ \forall i_j \in P_{orig}|_{i_1} \ \forall i_k \in P_{ext}|_{i_j}$  if  $\forall P \in \mathbf{P} \ i_k \notin P|_{i_1}$ , then  $\exists P'_{orig}$  such that  $P_{orig}|_{i_1} \cap P'_{orig}|_{i_1} \neq \emptyset$  and  $i_k \in P'|_{i_1}$  for some  $P' \in \text{extend}(P'_{orig}, P_{ext})$ .*

**Proof.** Proof by construction. Let  $\sigma$  be an arbitrary skeleton, and let  $i_1 \in \sigma(I_1)$ ,  $i_j \in P_{orig}|_{i_1}$ , and  $i_k \in P_{ext}|_{i_j}$  be arbitrary instances such that  $i_k \notin P|_{i_1}$  for any  $P \in \mathbf{P}$ .

Since  $i_j \in P_{orig}|_{i_1}$  and  $i_k \in P_{ext}|_{i_j}$ , but  $i_k \notin P|_{i_1}$ , there exists no pivot that yields a common subsequence in  $P_{orig}$  and  $P_{ext}$  that produces a path in  $\text{extend}$  that can reach  $i_k$ .

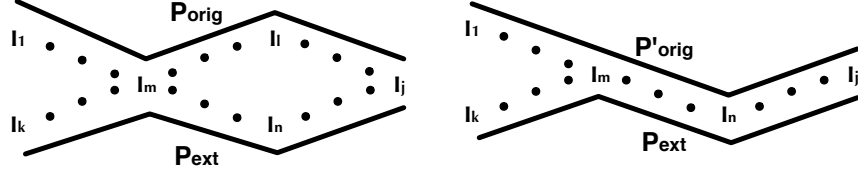


Figure 16: Schematic of the relational paths expected in Lemma 5.2. If item  $i_k$  is unreachable via  $extend(P_{orig}, P_{ext})$ , then there must exist a  $P'_{orig}$  of the form  $[I_1, \dots, I_m, \dots, I_n, \dots, I_j]$ .

Let the first divergent item class along the reverse of  $P_{orig}$  be  $I_l$  and along  $P_{ext}$  be  $I_n$ . The two paths must not only diverge, but they also necessarily reconverge at least once. If  $P_{orig}$  and  $P_{ext}$  do not reconverge, then there are no reoccurrences of an item class along any  $P \in \mathbf{P}$  that would restrict the inclusion of  $i_k$  in some terminal set  $P|_{i_1}$ . The sole reason that  $i_k \notin P|_{i_1}$  for any  $P \in \mathbf{P}$  is by Definition 4.4, through the bridge burning semantics of terminal sets.

Without loss of generality, assume  $P_{orig}$  and  $P_{ext}$  reconverge once, at item class  $I_m$ . So,  $P_{orig} = [I_1, \dots, I_m, \dots, I_l, \dots, I_j]$  and  $P_{ext} = [I_j, \dots, I_n, \dots, I_m, \dots, I_k]$  with  $I_l \neq I_n$ , as depicted in Figure 16. Let  $P'_{orig} = [I_1, \dots, I_m, \dots, I_n, \dots, I_j]$ , which is a valid relational path because  $[I_1, \dots, I_m]$  is a subpath of  $P_{orig}$  and  $[I_m, \dots, I_n, \dots, I_j]$  is a subpath of  $P_{ext}$ .

By construction,  $i_j \in P_{orig}|_{i_1} \cap P'_{orig}|_{i_1}$ . If  $P' = [I_1, \dots, I_m, \dots, I_k] \in extend(P'_{orig}, P_{ext})$  with pivot at  $I_m$ , then  $i_k \in P'|_{i_1}$ . ■

**Theorem 5.2** *For any acyclic relational model  $\mathcal{M}$ , perspective  $B \in \mathcal{E} \cup \mathcal{R}$ , and hop threshold  $h \in \mathbb{N}^0$ , the abstract ground graph  $AGG_{\mathcal{M}Bh}$  is  $(B, h)$ -reachably sound and complete for any ground graph  $GG_{\mathcal{M}\sigma}$  for all skeletons  $\sigma$ .*

**Proof.** Let  $\mathcal{M} = (\mathcal{S}, \mathcal{D})$  be an arbitrary acyclic relational model, let  $B \in \mathcal{E} \cup \mathcal{R}$  be an arbitrary perspective, and let  $h \in \mathbb{N}^0$  be an arbitrary hop threshold. We first show that  $AGG_{\mathcal{M}Bh}$  is  $(B, h)$ -reachably sound, and then prove that  $AGG_{\mathcal{M}Bh}$  is  $(B, h)$ -reachably complete. Figure 7 represents the general connection between abstract ground graphs and ground graphs.

(1) To prove that  $AGG_{\mathcal{M}Bh}$  is  $(B, h)$ -reachably sound, we must show that for every edge  $P_k.V_1 \rightarrow P_j.V_2$  in  $AGG_{\mathcal{M}Bh}$ , there exists a corresponding edge  $i_k.V_1 \rightarrow i_j.V_2$  in the ground graph  $GG_{\mathcal{M}\sigma}$  for some skeleton  $\sigma$  where  $i_k \in P_k|_b$  and  $i_j \in P_j|_b$  for some  $b \in \sigma(B)$ . There are three subcases, one for each type of edge in an abstract ground graph:

(a) Let  $[B, \dots, I_k].V_1 \rightarrow [B, \dots, I_j].V_2 \in RVE$  be an arbitrary edge in  $AGG_{\mathcal{M}Bh}$  between a pair of relational variables. Assume for contradiction that there exists no edge  $i_k.V_1 \rightarrow i_j.V_2$  connecting variable instances in any ground graph, that is,  $\forall b \in \sigma(B) \forall i_k.V_1 \in [B, \dots, I_k].V_1|_b \forall i_j.V_2 \in [B, \dots, I_j].V_2|_b \ i_k.V_1 \rightarrow i_j.V_2 \notin GG_{\mathcal{M}\sigma}$  for any skeleton  $\sigma$ . By Definition 5.2, if the abstract ground graph has edge  $[B, \dots, I_k].V_1 \rightarrow [B, \dots, I_j].V_2 \in RVE$ , then the model must have dependency  $[I_j, \dots, I_k].V_1 \rightarrow [I_j].V_2 \in \mathcal{D}$  and the relational

path  $[B, \dots, I_k] \in \text{extend}([B, \dots, I_j], [I_j, \dots, I_k])$ . So, by Definition 4.10 for ground graphs, there is an edge to every  $i_j.V_2$  from every  $i_k.V_1$  in the terminal set for  $i_j$  along  $[I_j, \dots, I_k].V_1$ , that is,  $\forall i_j \in \sigma(I_j) \forall i_k \in [I_j, \dots, I_k]|_{i_j} i_k.V_1 \rightarrow i_j.V_2 \in GG_{\mathcal{M}\sigma}$  for any skeleton  $\sigma$ . Since  $[B, \dots, I_k] \in \text{extend}([B, \dots, I_j], [I_j, \dots, I_k])$ , by Lemma 5.1 we know that there exists a skeleton  $\sigma$  such that  $\exists i_k \in [B, \dots, I_k]|_b \exists i_j \in [B, \dots, I_j]|_b$  such that  $i_k \in [I_j, \dots, I_k]|_{i_j}$  for some  $b \in \sigma(B)$ . So,  $\exists b \in \sigma(B) \exists i_k \in [B, \dots, I_k]|_b \exists i_j \in [B, \dots, I_j]|_b$  such that  $i_k.V_1 \rightarrow i_j.V_2 \in GG_{\mathcal{M}\sigma}$  for some skeleton  $\sigma$ , which contradicts the assumption.

(b) Let  $P_1.V_1 \cap P_2.V_1 \rightarrow [B, \dots, I_j].V_2 \in IVE$  be an arbitrary edge in  $AGG_{\mathcal{M}Bh}$  between an intersection variable and a relational variable, where  $P_1 = [B, \dots, I_m, \dots, I_k]$  and  $P_2 = [B, \dots, I_n, \dots, I_k]$  with  $I_m \neq I_n$ . By Lemma 4.1, there exists a skeleton  $\sigma$  such that  $P_1|_b \cap P_2|_b \neq \emptyset$  for some  $b \in \sigma(B)$ . Let  $i_k \in P_1|_b \cap P_2|_b$  for such a  $b \in \sigma(B)$  for  $\sigma$ . Assume for contradiction that  $\forall i_j \in [B, \dots, I_j]|_b i_k.V_1 \rightarrow i_j.V_2 \notin GG_{\mathcal{M}\sigma}$ . By Definition 5.2, if the abstract ground graph has edge  $P_1.V_1 \cap P_2.V_1 \rightarrow [B, \dots, I_j].V_2 \in IVE$ , then either  $P_1.V_1 \rightarrow [B, \dots, I_j].V_2 \in RVE$  or  $P_2.V_1 \rightarrow [B, \dots, I_j].V_2 \in RVE$ . Then, as shown in case (a),  $\exists i_j \in [B, \dots, I_j]|_b$  such that  $i_k.V_1 \rightarrow i_j.V_2 \in GG_{\mathcal{M}\sigma}$ , which contradicts the assumption.

(c) Let  $[B, \dots, I_k].V_1 \rightarrow P_1.V_2 \cap P_2.V_2 \in IVE$  be an arbitrary edge in  $AGG_{\mathcal{M}Bh}$  between a relational variable and an intersection variable, where  $P_1 = [B, \dots, I_m, \dots, I_j]$  and  $P_2 = [B, \dots, I_n, \dots, I_j]$  with  $I_m \neq I_n$ . The proof follows case (b) to show that  $\forall i_k \in [B, \dots, I_k]|_b \exists i_j \in P_1.V_2 \cap P_2.V_2|_b$  such that  $i_k.V_1 \rightarrow i_j.V_2 \in GG_{\mathcal{M}\sigma}$  for some skeleton  $\sigma$  and  $b \in \sigma(B)$  for which  $i_j \in P_1|_b \cap P_2|_b$ .

(2) To prove that the abstract ground graph  $AGG_{\mathcal{M}Bh}$  is  $(B, h)$ -reachably complete, we show that for every  $(B, h)$ -reachable edge  $i_k.V_1 \rightarrow i_j.V_2$  in any ground graph  $GG_{\mathcal{M}\sigma}$  for some skeleton  $\sigma$ , there is a corresponding edge  $X \rightarrow Y$  in  $AGG_{\mathcal{M}Bh}$ . Specifically, the  $(B, h)$ -reachable edge  $i_k.V_1 \rightarrow i_j.V_2$  yields two sets of relational variables for some  $b \in \sigma(B)$ , namely  $\mathbf{P}_k.V_1 = \{P_k.V_1 \mid i_k \in P_k|_b \wedge \text{length}(P_k) \leq h+1\}$  and  $\mathbf{P}_j.V_2 = \{P_j.V_2 \mid i_j \in P_j|_b \wedge \text{length}(P_j) \leq h+1\}$  by Definition 5.4. Note that all relational variables in both  $\mathbf{P}_k.V_1$  and  $\mathbf{P}_j.V_2$  are nodes in  $AGG_{\mathcal{M}Bh}$ , as are all pairwise intersection variables:  $\forall P_k.V_1, P'_k.V_1 \in \mathbf{P}_k.V_1 \ P_k.V_1 \cap P'_k.V_1 \in AGG_{\mathcal{M}Bh}$  and  $\forall P_j.V_2, P'_j.V_2 \in \mathbf{P}_j.V_2 \ P_j.V_2 \cap P'_j.V_2 \in AGG_{\mathcal{M}Bh}$ .

Let  $\sigma$  be an arbitrary skeleton, and let  $i_k.V_1 \rightarrow i_j.V_2 \in GG_{\mathcal{M}\sigma}$  be an arbitrary  $(B, h)$ -reachable edge drawn from  $[I_j, \dots, I_k].V_1 \rightarrow [I_j].V_2 \in \mathcal{D}$  where such that  $\mathbf{P}_k.V_1 \neq \emptyset$  and  $\mathbf{P}_j.V_2 \neq \emptyset$ . We show that  $\forall P_k.V_1 \in \mathbf{P}_k.V_1 \forall P_j.V_2 \in \mathbf{P}_j.V_2$  either (a)  $P_k.V_1 \rightarrow P_j.V_2 \in AGG_{\mathcal{M}Bh}$ , (b)  $P_k.V_1 \cap P'_k.V_1 \rightarrow P_j.V_2 \in AGG_{\mathcal{M}Bh}$ , where  $P'_k.V_1 \in \mathbf{P}_k.V_1$ , or (c)  $P_k.V_1 \rightarrow P_j.V_2 \cap P'_j.V_2 \in AGG_{\mathcal{M}Bh}$ , where  $P'_j.V_2 \in \mathbf{P}_j.V_2$ . Let  $P_k.V_1 \in \mathbf{P}_k.V_1, P_j.V_2 \in \mathbf{P}_j.V_2$  be an arbitrary pair of relational variables drawn from the sets  $\mathbf{P}_k.V_1$  and  $\mathbf{P}_j.V_2$ .

(a) If  $P_k \in \text{extend}(P_j, [I_j, \dots, I_k])$ , then  $P_k.V_1 \rightarrow P_j.V_2 \in AGG_{\mathcal{M}Bh}$  by Definition 5.2.

(b) If  $P_k \notin \text{extend}(P_j, [I_j, \dots, I_k])$ , but  $\exists P'_k \in \text{extend}(P_j, [I_j, \dots, I_k])$  such that  $P'_k.V_1 \in \mathbf{P}_k.V_1$ , then  $P'_k.V_1 \rightarrow P_j.V_2 \in AGG_{\mathcal{M}Bh}$ , and  $P_k.V_1 \cap P'_k.V_1 \rightarrow P_j.V_2 \in AGG_{\mathcal{M}Bh}$  by Definition 5.2.

(c) If  $\forall P \in \text{extend}(P_j, [I_j, \dots, I_k]) P.V_1 \notin \mathbf{P}_k.V_1$ , then, by Lemma 5.2,  $\exists P'_j$  such that  $i_j \in P'_j|_b$  and  $P_k \in \text{extend}(P'_j, [I_j, \dots, I_k])$ . So,  $P'_j.V_2 \in \mathbf{P}_j.V_2$ ,  $P_k.V_1 \rightarrow P'_j.V_2 \in AGG_{\mathcal{M}Bh}$ , and  $P_k.V_1 \rightarrow P'_j.V_2 \cap P_j.V_2 \in AGG_{\mathcal{M}Bh}$  by Definition 5.2. ■



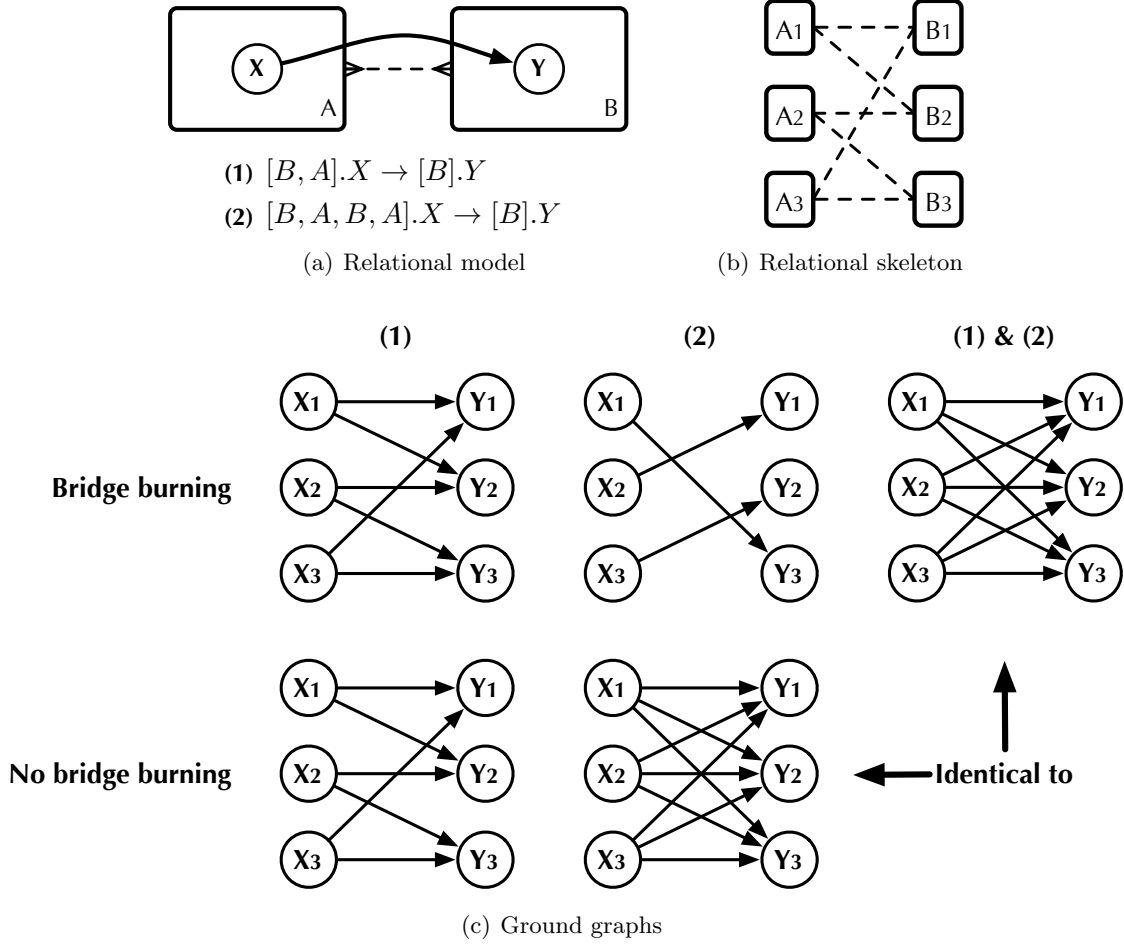


Figure 17: Example demonstrating that bridge burning semantics yields a more expressive class of models than semantics without bridge burning. (a) Relational model over a schema with two entity classes and two attributes with two possible relational dependencies (relationship class omitted for simplicity). (b) Simple relational skeleton with three  $A$  and three  $B$  instances. (c) Bridge burning semantics yields three possible ground graphs with combinations of dependencies (1) and (2) whereas no bridge burning yields two possible ground graphs. The bridge burning ground graphs subsume the no bridge burning ground graphs.

## Appendix B.

In this appendix, we provide an example to show that the bridge burning semantics for terminal sets of relational paths yields a strictly more expressive class of relational models than semantics without bridge burning. The bridge burning semantics produces terminal sets that are necessarily *subsets* of terminal sets which would otherwise be produced without bridge burning. Paradoxically, this enables a *superset* of relational models.

Recall the definition of a terminal set for a relational path:

**Definition 4.4 (Terminal set of a relational path)** For any skeleton  $\sigma_{\mathcal{ER}}$  and any  $i_1 \in \sigma(I_1)$ , a *terminal set*  $P|_{i_1}$  for relational path  $P = [I_1, \dots, I_k]$  can be defined inductively as

$$[I_1]|_{i_1} = \{i_1\}$$

$$[I_1, \dots, I_{k-1}, I_k]|_{i_1} = \bigcup_{i_{k-1} \in [I_1, \dots, I_{k-1}]|_{i_1}} \{i_k \mid ((i_{k-1} \in i_k \text{ if } I_k \in \mathcal{R}) \vee (i_k \in i_{k-1} \text{ if } I_k \in \mathcal{E})) \wedge i_k \notin [I_1, \dots, I_j]|_{i_1} \text{ for } j = 1 \text{ to } k-1\}$$

The final condition in the inductive definition ( $i_k \notin [I_1, \dots, I_j]|_{i_1}$  for  $j = 1$  to  $k-1$ ) encodes bridge burning. The item  $i_k$  is only added to the terminal set if it is not a member of the terminal set of any previous subpath. For example, let  $P$  be the relational path  $[\text{EMPLOYEE}, \text{DEVELOPS}, \text{PRODUCT}, \text{DEVELOPS}, \text{EMPLOYEE}]$ . This relational path produces terminal sets that include the employees that work on the same products: co-workers. Instantiating this path with the employee Quinn,  $P|_{\text{Quinn}}$ , produces the terminal set  $\{\text{Paul}, \text{Roger}, \text{Sally}\}$ . Since  $\text{Quinn} \in [\text{EMPLOYEE}]|_{\text{Quinn}}$ , the bridge burning semantics excludes Quinn from this set. This makes intuitive sense as well—Quinn should not be considered her own colleague.

A relational model is simply a collection of relational dependencies. Each relational dependency is primarily described by the relational path of the parent relational variable (because, for canonically specified dependencies, the relational path of the child consists of a single item class). The relational path specification is used in the construction of ground graphs, connecting variable instances that appear in the terminal sets of the parent and child relational variables.

To characterize the expressiveness of relational models, we can inspect the space of representable ground graphs by choosing an arbitrary relational skeleton and a small set of relational dependencies. We show with a simple example that the bridge burning semantics for a model over a two-entity, bivariate schema yields more possible ground graphs than without bridge burning. (We omit the relationship class for simplicity.) In Figure 17(a), we present such a model with two possible relational dependencies labeled (1) and (2). Figure 17(b) provides a very simple relational skeleton involving three  $A$  and three  $B$  instances (relationship instances are represented as dashed lines for simplicity). As shown in Figure 17(c), the bridge burning semantics leads to three possible ground graphs, one for each combination of the dependencies (1), (2), and both (1) and (2) together. Without bridge burning, only two ground graphs are possible because dependency (2) completely subsumes dependency (1) with those semantics.

This example generalizes to arbitrary dependencies. The terminal sets of relational paths that repeat item classes subsume subpaths under no bridge burning semantics. This leads to fewer possible relational models, which justifies our choice of semantics for terminal sets of relational paths.

## Appendix C.

In this appendix, we provide additional details for the experiment in Section 7.1. The goal of this experiment is to determine which factors influence the size of abstract ground

Predictor	Coefficient	Partial	Semipartial
# relationships	1.15	0.451	0.153
# MANY cardinalities $\times$ isEntity=F	1.11	0.359	0.128
# entities	-0.74	0.352	0.101
# MANY cardinalities $\times$ isEntity=T	0.91	0.217	0.069
# MANY cardinalities $\times$ # relationships	-0.33	0.108	0.022
# attributes	0.08	0.024	0.005

Table 1: Estimated standardized coefficient, squared partial correlation coefficient, and squared semipartial correlation coefficient for each predictor in the log-linear model of the number of nodes in an abstract ground graph.

graphs because the computational complexity of relational  $d$ -separation depends on its size. Specifically, we show here the results of running log-linear regression to predict the size of abstract ground graphs for varying schemas and models. We standardized the input variables by dividing by two standard deviations, as recommended by Gelman (2008). Since the predictor for the number of dependencies is log-transformed, the standardization for that variable occurs after taking the logarithm.

In predicting the (log of the) number of nodes, the following variables were significantly and substantively predictive (in order of decreasing predictive power):

- Number of relationships (positive)
- Interaction between MANY cardinalities and an indicator variable for whether the AGG is from an entity or relationship perspective (positive)
- Number of entities (negative)
- Interaction between the number of MANY cardinalities and relationships (negative)
- Total number of attributes (positive)

The fit for the nodes model has an  $R^2 = 0.814$  for  $n = 449,993$ , and Table 1 contains the standardized coefficients as well as the squared partial and semipartial correlation coefficients for each predictor. In predicting the (log of the) number of edges, the following variables were significantly and substantively predictive (in order of decreasing predictive power):

- Log of the number of dependencies (positive)
- Number of relationships (positive)
- Interaction between MANY cardinalities and an indicator variable for whether the AGG is from an entity or relationship perspective (positive)
- Number of entities (negative)
- Interaction between the number of MANY cardinalities and relationships (negative)

Predictor	Coefficient	Partial	Semipartial
$\log(\# \text{ dependencies})$	0.41	0.440	0.165
$\# \text{ relationships}$	1.09	0.395	0.138
$\# \text{ MANY cardinalities} \times \text{isEntity=F}$	1.21	0.356	0.135
$\# \text{ entities}$	-0.78	0.353	0.115
$\# \text{ MANY cardinalities} \times \text{isEntity=T}$	1.00	0.231	0.077
$\# \text{ MANY cardinalities} \times \# \text{ relationships}$	-0.38	0.127	0.031

Table 2: Estimated standardized coefficient, squared partial correlation coefficient, and squared semipartial correlation coefficient for each predictor in the log-linear model of the number of edges in an abstract ground graph.

The fit for the edges model has an  $R^2 = 0.789$  for  $n = 449,993$ , and Table 2 contains the standardized coefficients and the squared partial and semipartial correlation coefficients for each predictor.

## References

- Richard Barker. *CASE Method: Entity Relationship Modeling*. Addison-Wesley, Boston, MA, 1990.
- Wray L. Buntine. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.
- Nir Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659):799–805, 2004.
- Dan Geiger and Judea Pearl. On the logic of causal models. In *Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*, pages 136–147, 1988.
- Dan Geiger, Thomas Verma, and Judea Pearl. Identifying independence in Bayesian networks. *Networks*, 20(5):507–534, 1990.
- Andrew Gelman. Scaling regression inputs by dividing by two standard deviations. *Statistics in Medicine*, 27(15):2865–2873, 2008.
- Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multi-level/Hierarchical Models*. Cambridge University Press New York, 2007.
- Lise Getoor. *Learning Statistical Models from Relational Data*. Ph.D. thesis, Stanford University, 2001.
- Lise Getoor and Ben Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA, 2007.
- Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Ben Taskar. Probabilistic relational models. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, pages 129–174. MIT Press, Cambridge, MA, 2007.

- W. R. Gilks, A. Thomas, and D. J. Spiegelhalter. A language and program for complex bayesian modeling. *The Statistician*, 43:169–177, 1994.
- David Heckerman, Chris Meek, and Daphne Koller. Probabilistic entity-relationship models, PRMs, and plate models. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, pages 201–238. MIT Press, Cambridge, MA, 2007.
- Michael G. Hudgens and M. Elizabeth Halloran. Toward causal inference with interference. *Journal of the American Statistical Association*, 103(482):832–842, 2008.
- Stefan Kramer, Nada Lavrač, and Peter Flach. Propositionalization approaches to relational data mining. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 262–286. Springer-Verlag, New York, NY, 2001.
- Marc Maier, Brian Taylor, Hüseyin Oktay, and David Jensen. Learning causal models of relational domains. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.
- Richard E. Neapolitan. *Learning Bayesian Networks*. Pearson Prentice Hall Upper Saddle River, NJ, 2004.
- Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, 2000.
- Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Fransico, CA, 1988.
- Richard Scheines. An introduction to causal inference. In Vaughan R. McKim and Steven P. Turner, editors, *Causality in Crisis? Statistical Methods and the Search for Causal Knowledge in the Social Sciences*, pages 185–199. University of Notre Dame Press, 1997.
- Eran Segal, Ben Taskar, Audrey Gasch, Nir Friedman, and Daphne Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(suppl 1):S243–S252, 2001.
- Cosma R. Shalizi and Andrew C. Thomas. Homophily and contagion are generically confounded in observational social network studies. *Sociological Methods & Research*, 40(2): 211–239, 2011.
- Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction and Search*. MIT Press, Cambridge, MA, 2nd edition, 2000.
- Ben Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 870–878, 2001.
- Eric J. Tchetgen Tchetgen and Tyler J. VanderWeele. On causal inference in the presence of interference. *Statistical Methods in Medical Research*, 21(1):55–75, 2012.
- Jin Tian, Azaria Paz, and Judea Pearl. *Finding Minimal D-separators*. Technical Report R-254, UCLA Computer Science Department, February 1998.

Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, October 2006.

Thomas Verma and Judea Pearl. Causal networks: Semantics and expressiveness. In *Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*, pages 352–359, 1988.